

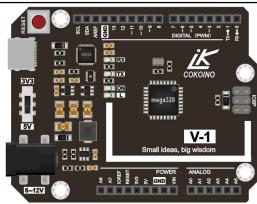
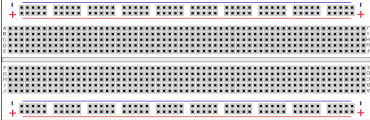

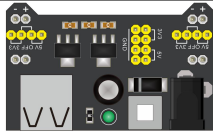







13. Music keyboard

ABOUT THIS PROJECT:

You will learn:

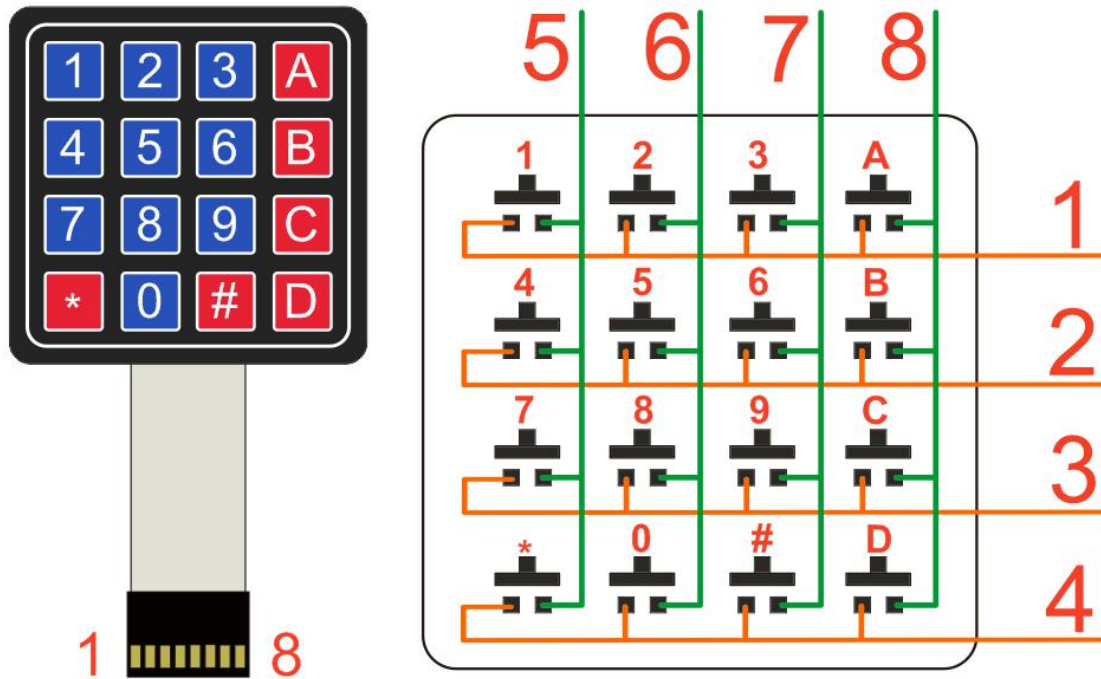
◆ How to make a music keyboard

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
30 CM 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. 4 X 4 Membrane Matrix keypad

4 X 4 Membrane Matrix keypad has 8 pins, which are divided into columns and rows, as shown below:



How it Works

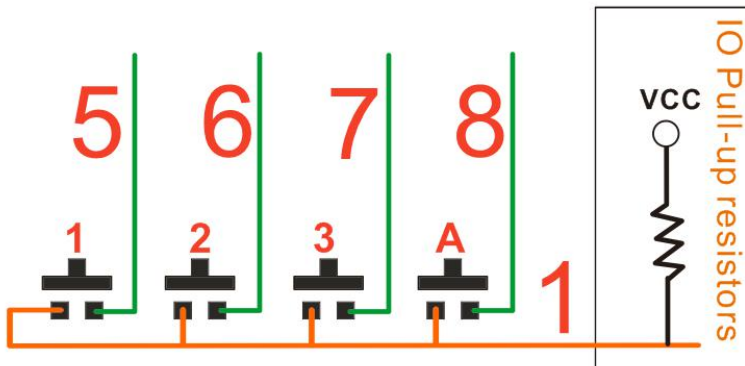
Matrix keypads use a combination of four rows and four columns to provide button states to the host device, typically a microcontroller. Underneath each key is a pushbutton, with one end connected to one row, and the other end connected to one column.

These connections are shown above.

In order for the microcontroller to determine which button is pressed, it first needs to pull each of the four columns (pins 1-4) either low or high one at a time, and then poll the states of the four rows (pins 5-8). Depending on the states of the columns, the microcontroller can tell which button is pressed. For example, say your program pulls all four columns low and then pulls the first row high. It then reads the input states of each column, and reads pin 1 high. This means that a contact has been made between column 4 and row 1, so button 'A' has been pressed.

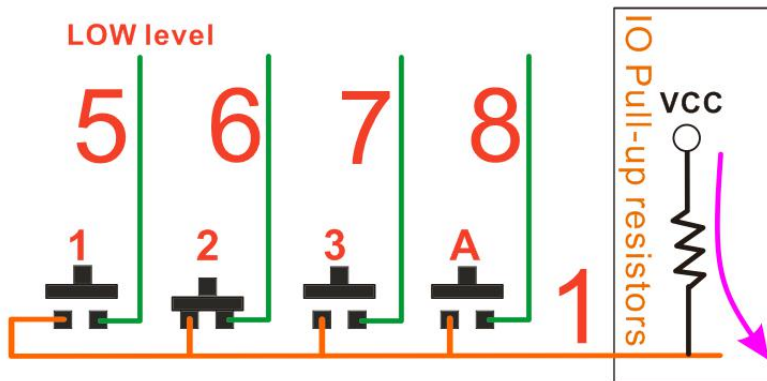
Matrix keypad detection and scanning principle

Simplify 4 × 4 keypad into 4 × 1 keypad, connect them to an input pin of the microcontroller

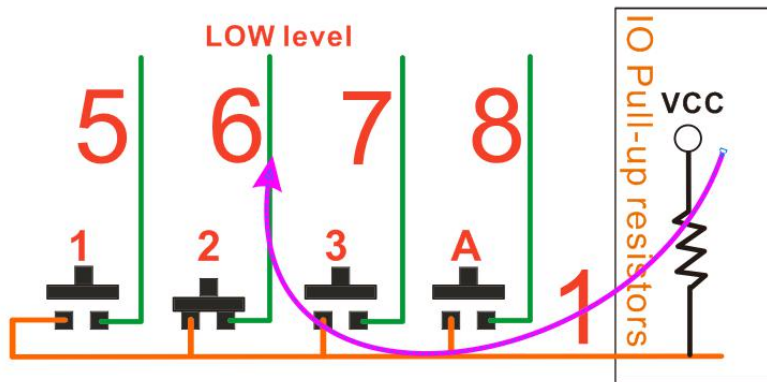


Assuming that if pin 5-8 all input high, whether the button is pressed or not, the Arduino board will get a high level. In order to detect whether the button is pressed, we need to use the arduino program to set the 5--8 pin to a low potential in sequence.

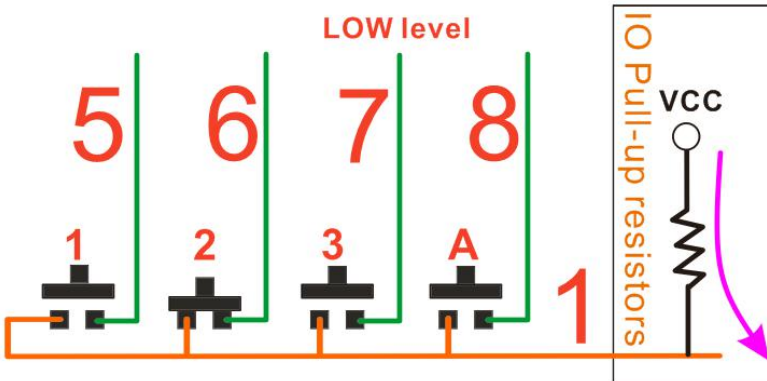
1) When pin 5 inputs a low level, pin 1 still get a high level, button 1 is not pressed



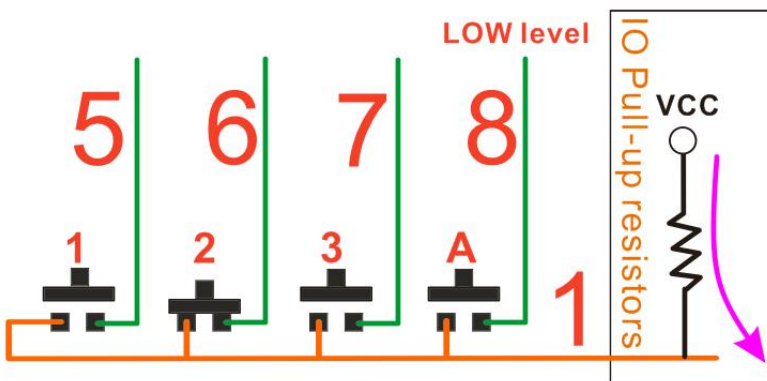
2) When pin 6 inputs a low level, pin 1 gets a low level, button 2 is pressed



3) When pin 7 inputs a low level, pin 1 still get a high level, button 3 is not pressed

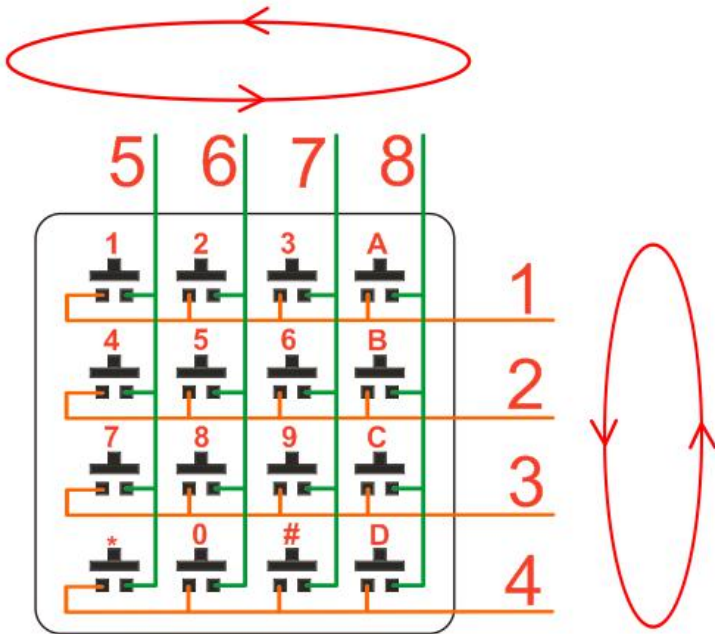


4) When pin 8 inputs a low level, pin 1 still get a high level, button A is not pressed



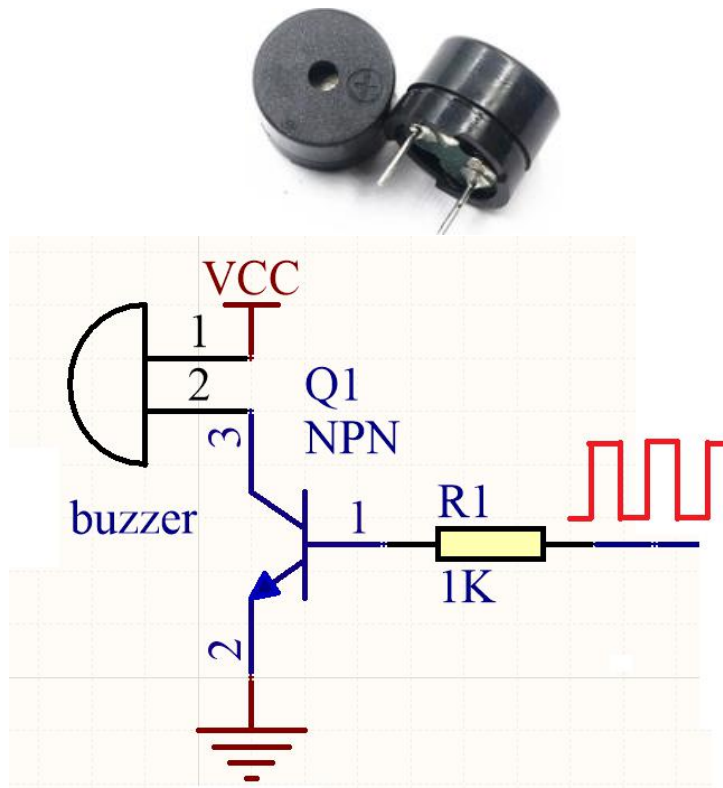
The arduino program needs to be executed repeatedly in order to continuously detect whether a key is pressed.

The process of detecting which key on the 4 * 4 keyboard is pressed requires a double loop to scan each column in batches:



3. Introducing passive buzzer

The passive buzzer does not have an oscillation source inside, and direct current cannot make it sound. It must be driven by a 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. When doing a project, generally connect its "+" sign to the positive pole of the circuit, and the other end to the negative pole of the circuit, as shown below:



4. Let's make a music keyboard

This experiment uses the keypad library function to read the key value of the 4*4 keypad, reducing the difficulty of writing the code, so please **copy the keypad library file to the libraries folder in the IDE installation directory**.

Then the key value read by V-1 board is used to make the microphone emit sounds of different frequencies and sizes through the PWM signal.

The main statement of this program is: PWM_data = analogRead (A0); analogWrite (pin, PWM_data); PMW signal frequency adjustment, and the use of IDE serial monitor.

For more information on PWM tuning, please refer to the following page:

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

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

3.1、Sketch

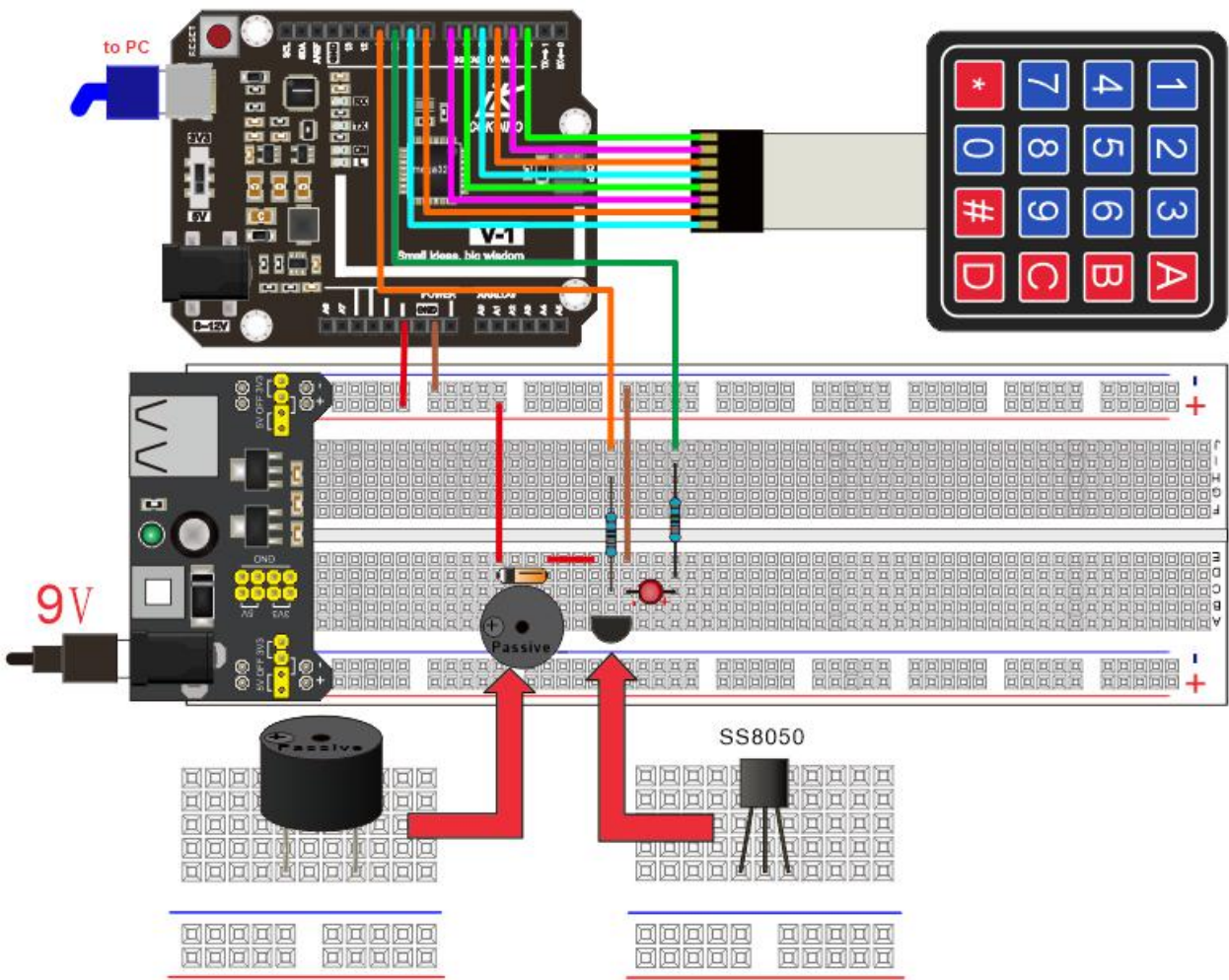
```
#include <Keypad.h>
#define buzzer 11
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the symbols on the buttons of the keypads
char hexaKeys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {2, 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);
  }
}
```

```

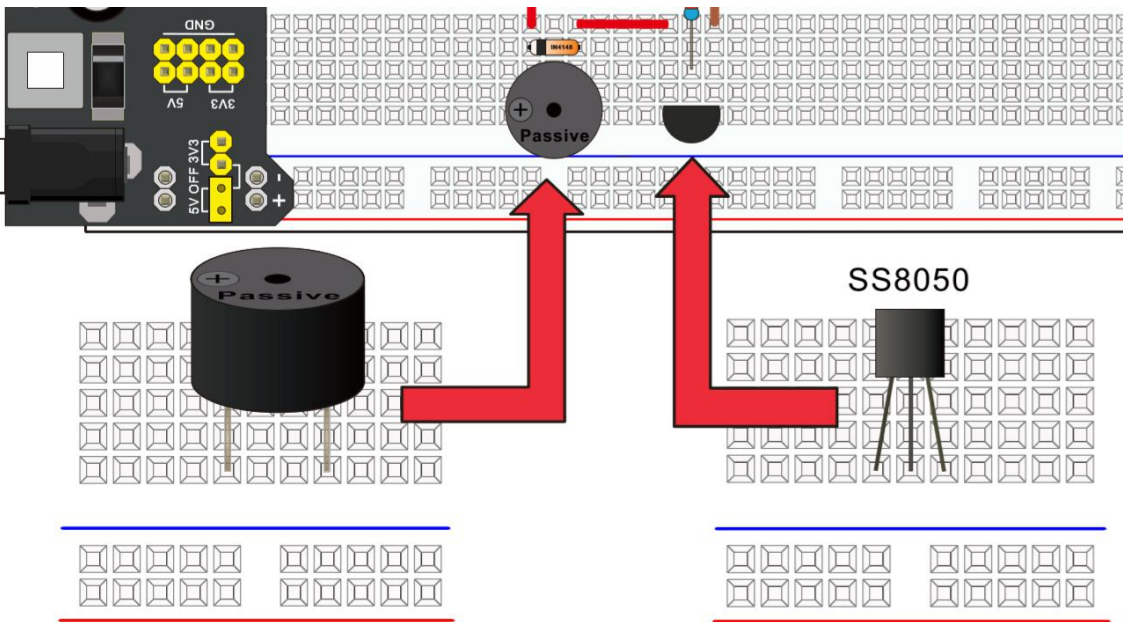
Serial.println(customKey);
}
}
/*
* Set pin 9's PWM frequency to 3906 Hz (31250/8 = 3906)
* Note that the base frequency for pins 3, 9, 10, and 11 is 31250 Hz
*setPwmFrequency(9, 8);
* Set pin 6's PWM frequency to 62500 Hz (62500/1 = 62500)
* Note that the base frequency for pins 5 and 6 is 62500 Hz
*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 = 0x07; break;
            default: return;
        }
        TCCR2B = TCCR2B & 0b11111000 | mode;
    }
}
}

```


3.2 Wiring Diagram

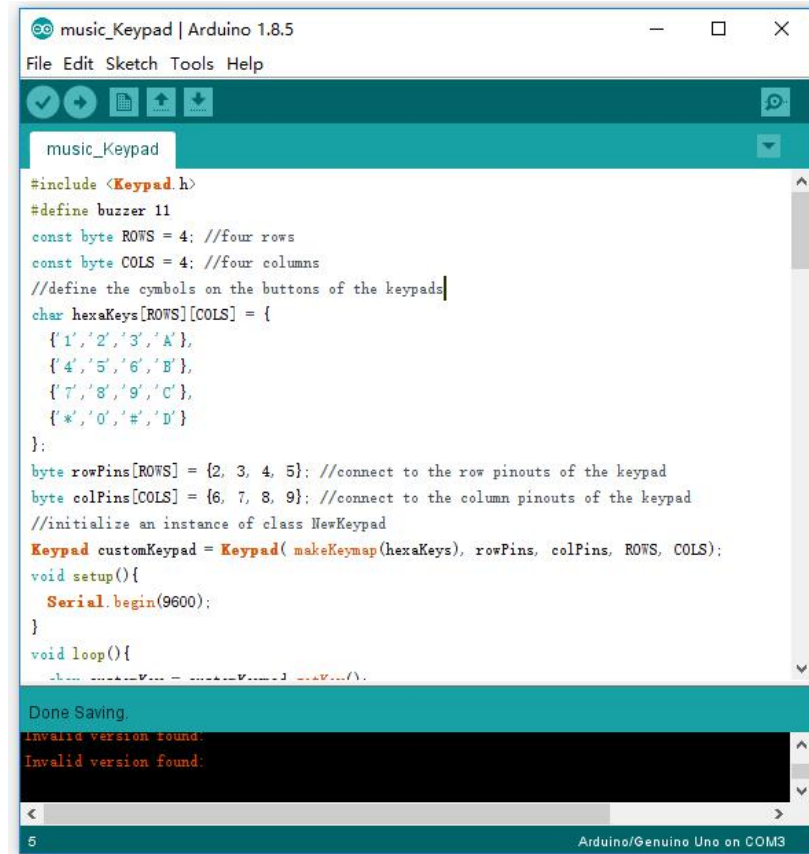


Detail enlargement



3.3 Steps

Connect the computer and V-1 board with a USB cable and copy the above sample code to the Arduino IDE as shown below:



```
#include <Keypad.h>

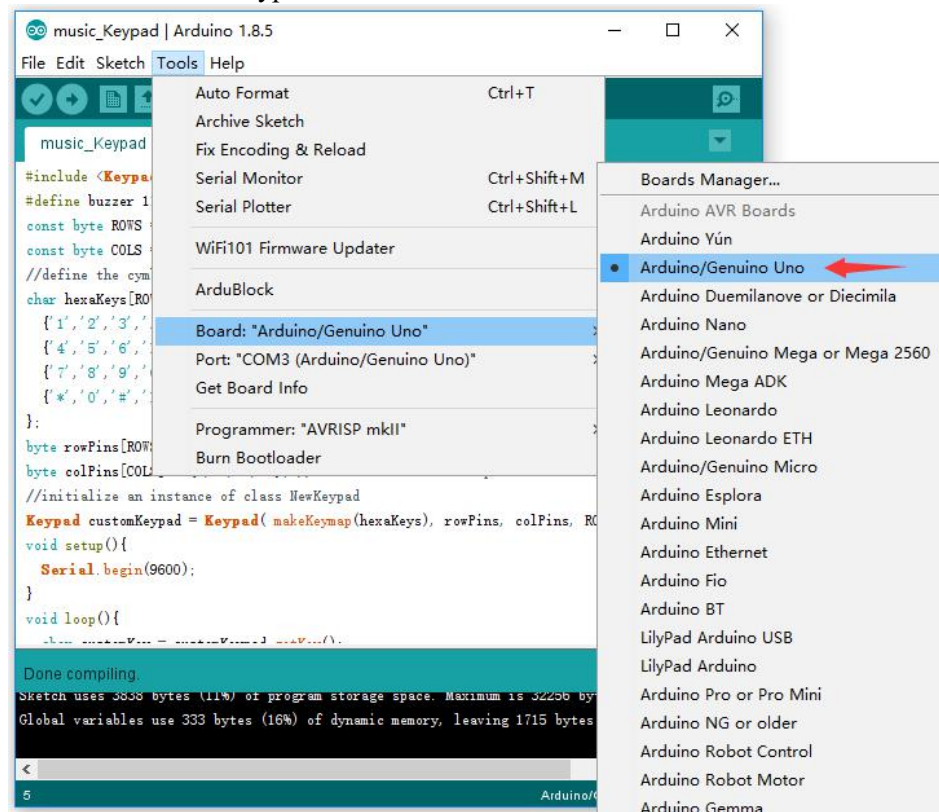
#define buzzer 11
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the symbols on the buttons of the keypad
char hexaKeys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

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);

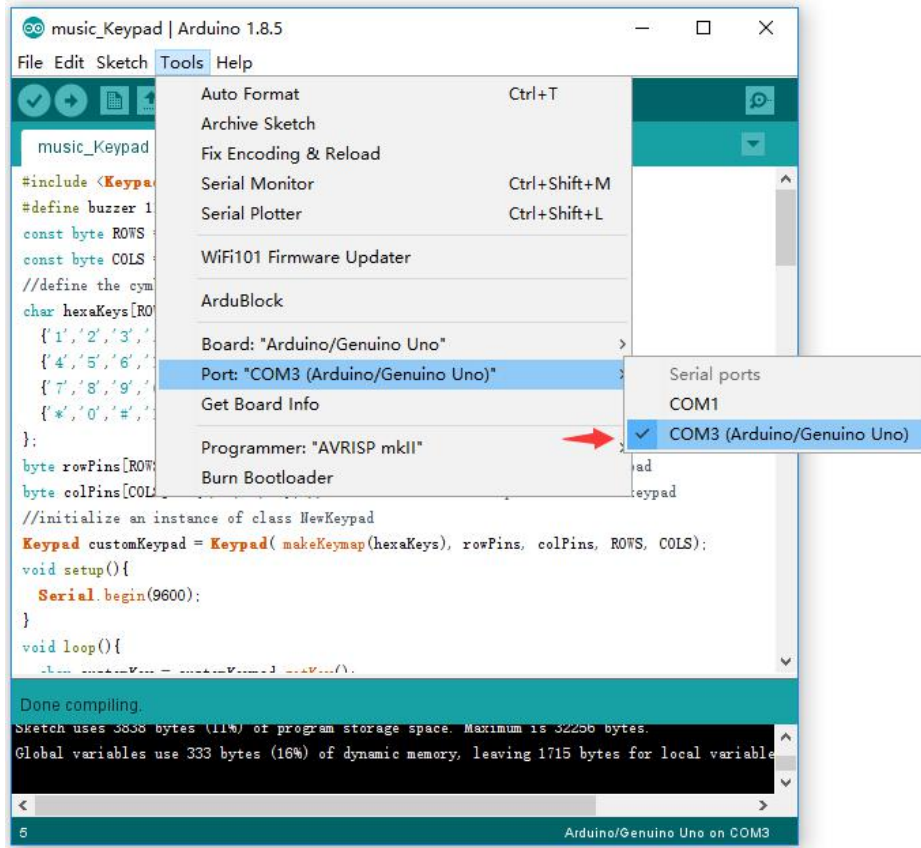
void setup(){
  Serial.begin(9600);
}

void loop(){
  //when customKeypad = customKeypad J useV...()
}
```

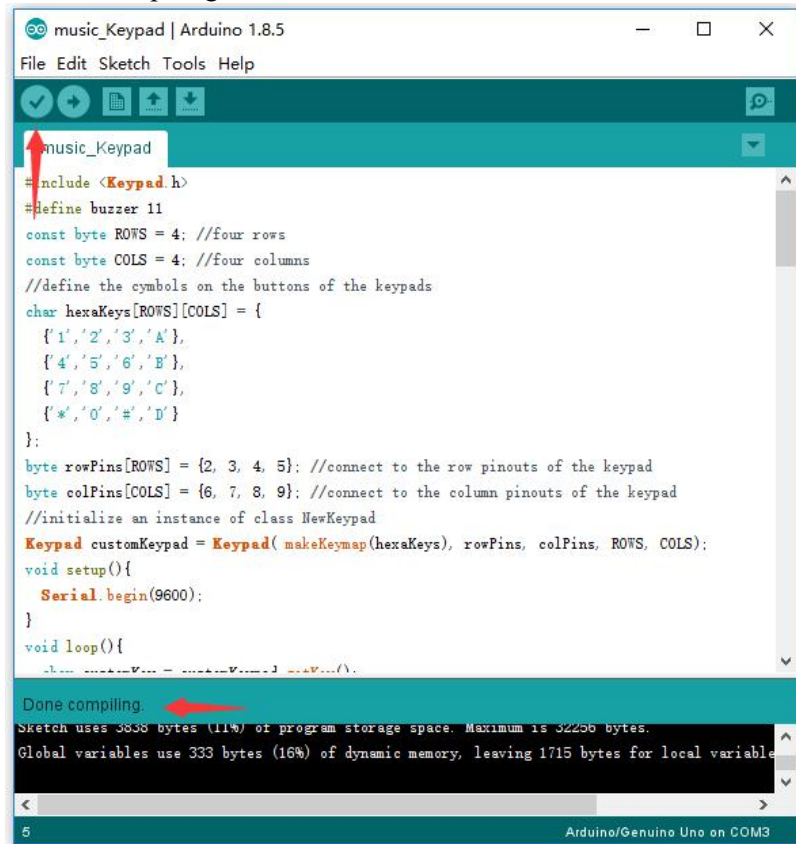
3.3.2、Select board type



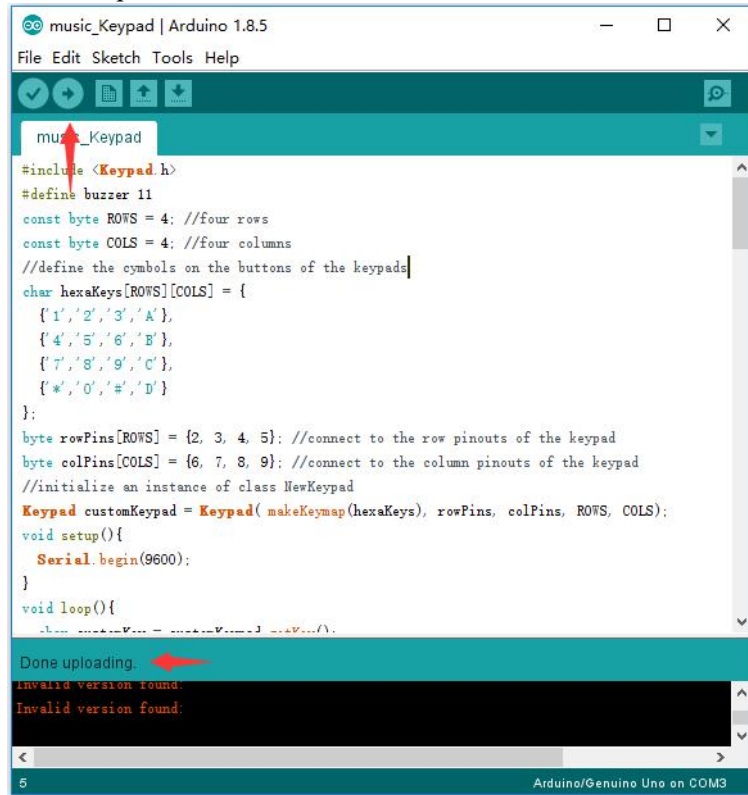
3.3.3、Select port



3.3.4、Compiling



3.3.5、Upload the sketch



3.3.6、Result

Unplug the USB cable from the V-1 board, connect the power module to the external power supply, and then turn on the switch of the power module on the breadboard. Open the IDE serial port monitor, adjust the baud rate to 9600, press the button on the 4*4 matrix keyboard on the hand, the buzzer will sound the sound of different size and frequency, and the serial monitor will print the key value of the corresponding button board. As shown below.

