

Lesson 3 Test the onboard passive buzzer

Table

1 Introduction of the buzzer	1
2 Principle.....	1
3 Components & Parts	3
4 Circuit.....	3
5 Upload code and test.....	4
6 Extended experiment	9
7 Any questions and suggestions are welcome	11

1 Introduction of the buzzer

Buzzer is a kind of voice device that converts audio model into sound signal. It is masterly used to prompt or alarm. According to different design and application, it can produce music sound, flute sound, buzzer, alarm sound, electric bell and other different sounds.

There are two types of piezoelectric buzzers that are commonly used in electronics projects – active buzzers and passive buzzers. Active buzzers are called active because they only need a DC voltage to produce sound. Passive buzzers need an AC voltage to produce sound.

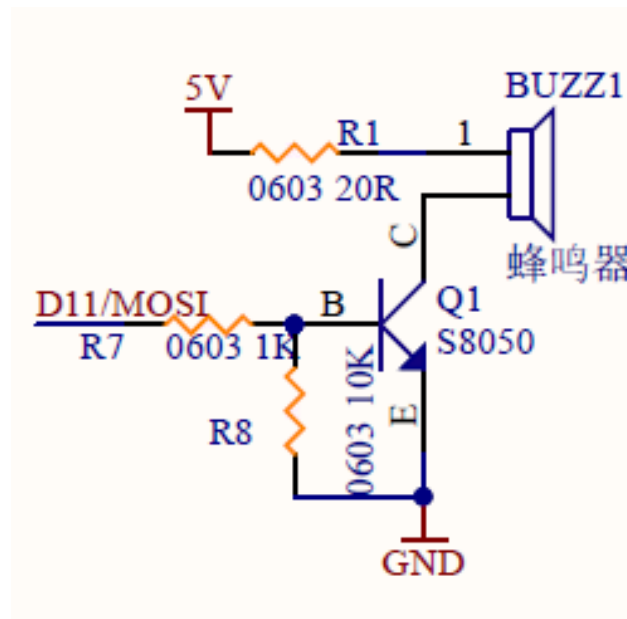
2 Principle

Active Buzzer doesn't need a varying voltage, you just give it constant voltage and it has an internal chopper (oscillator) circuit that turns the constant voltage to a square wave and feeds that to the piezo disc to generate a fixed-frequency tone.

Our onboard buzzer is a passive buzzer. Passive Buzzers need an AC voltage signal in order to vibrate and make sound. Therefore, it's a little bit harder to control compared to Active buzzers. But it also allows us to generate whatever note frequency we want to play.

Compared to the fixed tone that active buzzers generate, passive buzzers are more suitable for projects where you need to control the audio tone frequency.

Schematic:



This lesson controls the passive buzzer as a square wave signal, which occupies the D11 pin of Atmega328p on the control board.

Tutorial: <https://github.com/cokoino/CKK0002>

Download Tutorial :

Click the "Code" button, then click "Download ZIP" button in the pop-up window. Do NOT click the "Open with GitHub Desktop" button, it will lead you to install Github software.


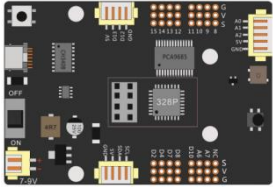
! Unzip the ZIP file instead of opening the file in the ZIP file directly.

! Do not move, delete or rename files in the folder just unzipped.

Save Tutorial :

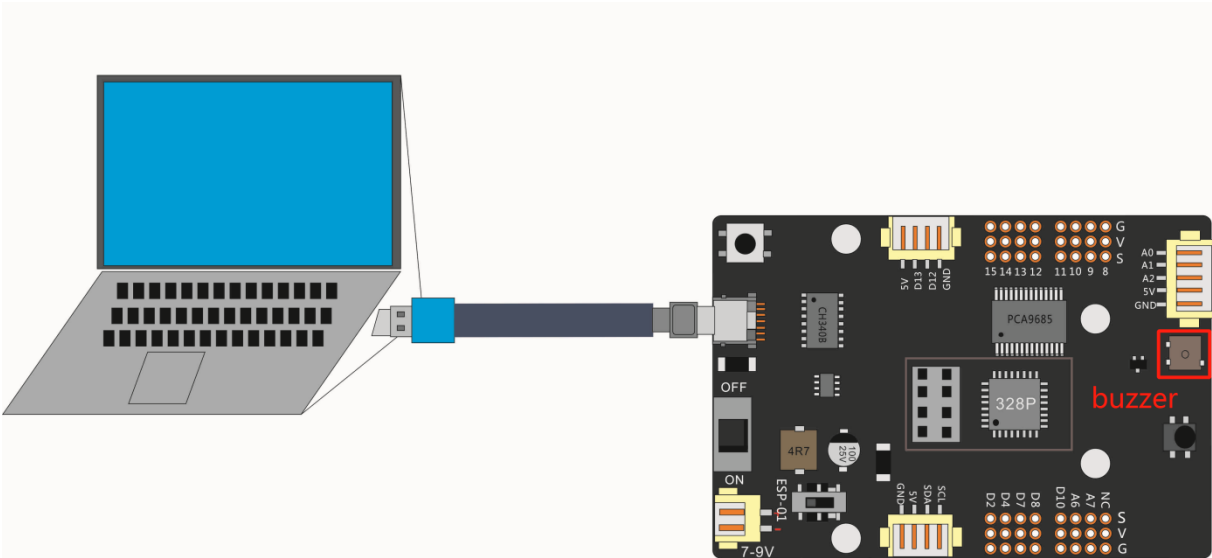
For the convenience of using and querying tutorials in the future, it is recommended to put the downloaded files in a local folder on the computer. For example, we put the files in E:\CKK0002-master

3 Components & Parts

Components	Quantity	Picture
USB cable	1	
Control board	1	

4 Circuit

Referring to the figure below, connect the control board to the computer with a USB cable. Since the power of the onboard buzzer is very small, it can be powered by a USB cable and does not require an external power supply.



5 Upload code and test

The code used in this lesson is placed in this folder:

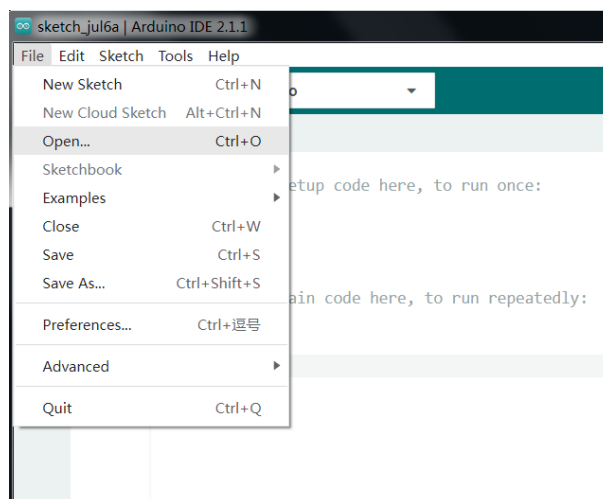
<E:\CKK0002-master\Tutorial\sketches>

5.1 Double-click the Arduino IDE shortcut on the desktop to open it



Buzzer_test

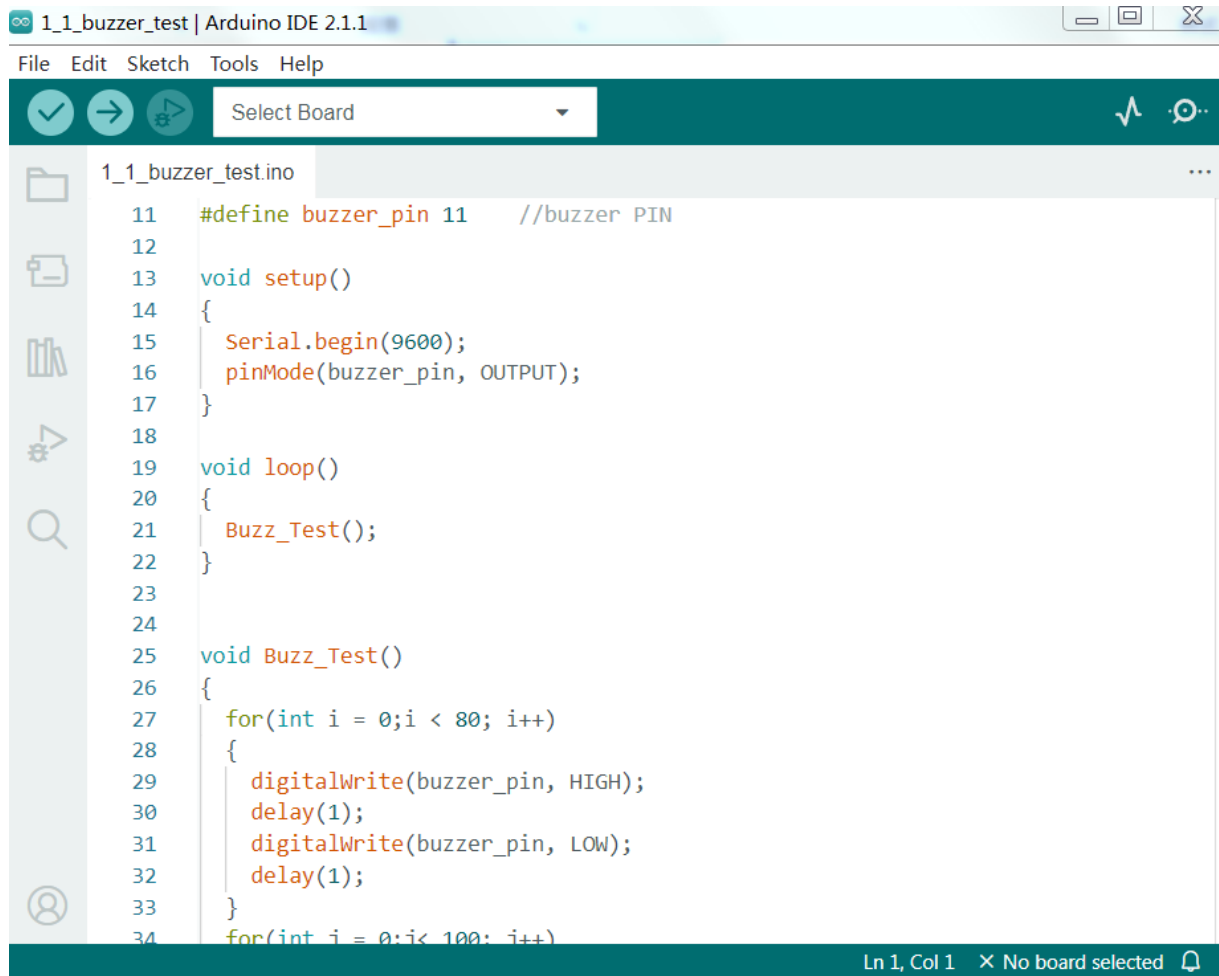
5.2 On the Arduino IDE interface, click "File" --- "open"



5.3 Select the code in the folder named 1_1_buzzer_test:

E:\CKK0002-master\Tutorial\sketches\1_1_buzzer_test

After selecting the code, click "open", the interface is as Follows

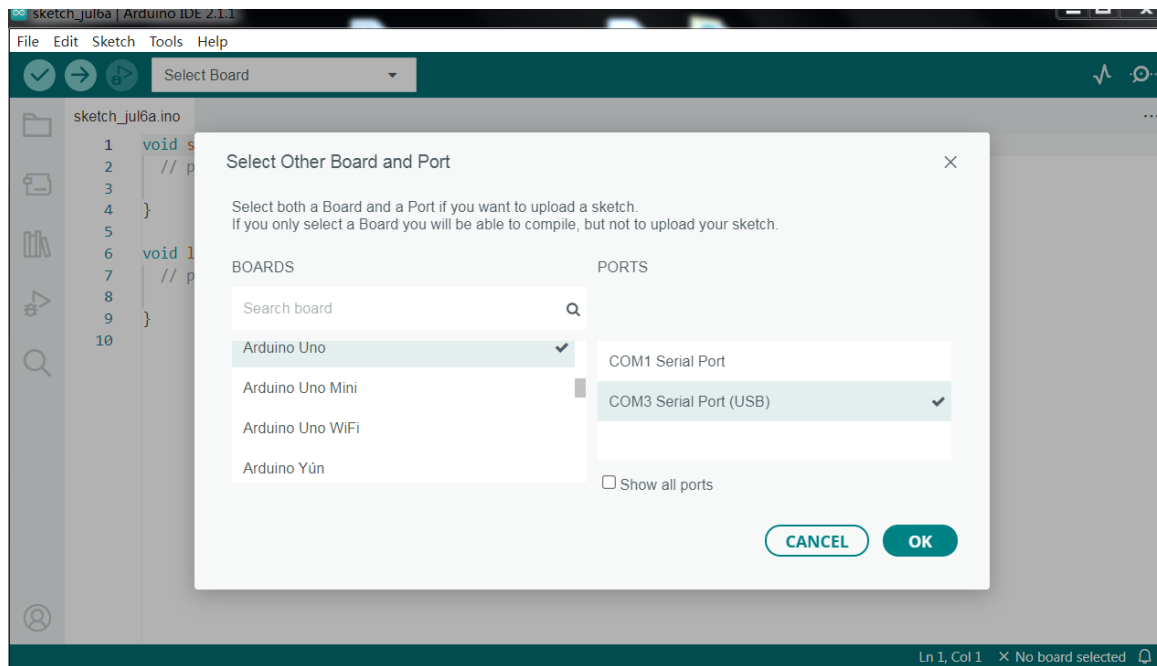


```
11 #define buzzer_pin 11 //buzzer PIN
12
13 void setup()
14 {
15     Serial.begin(9600);
16     pinMode(buzzer_pin, OUTPUT);
17 }
18
19 void loop()
20 {
21     Buzz_Test();
22 }
23
24
25 void Buzz_Test()
26 {
27     for(int i = 0; i < 80; i++)
28     {
29         digitalWrite(buzzer_pin, HIGH);
30         delay(1);
31         digitalWrite(buzzer_pin, LOW);
32         delay(1);
33     }
34     for(int i = 0; i < 100; i++)
```


5.4 Select board

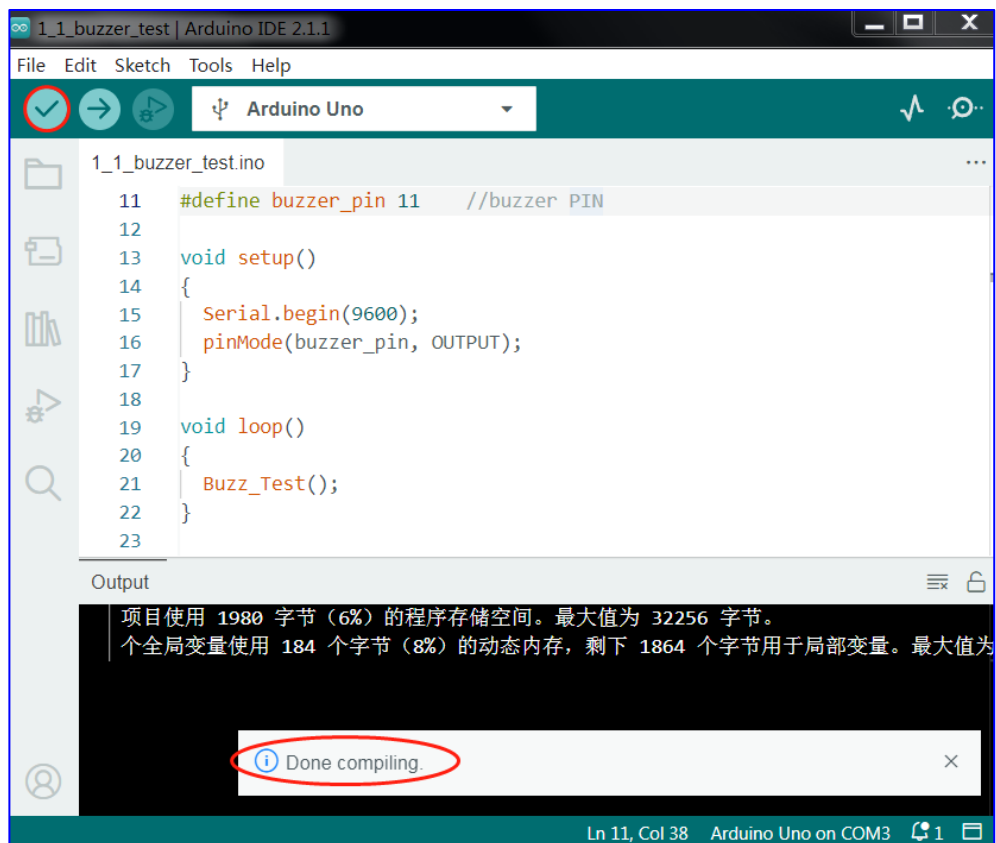
Click "[Select Board](#)", select the board as "[Arduino UNO](#)" in the pop-up "[Select Other Board and Port](#)" drop-down box,

Select PORTS as "[COM3 Serial Port](#)" (port is the port number of the board recognized by the computer, the port number recognized by each computer may be different, and the port recognized in this tutorial is COM3)




5.5 Compile

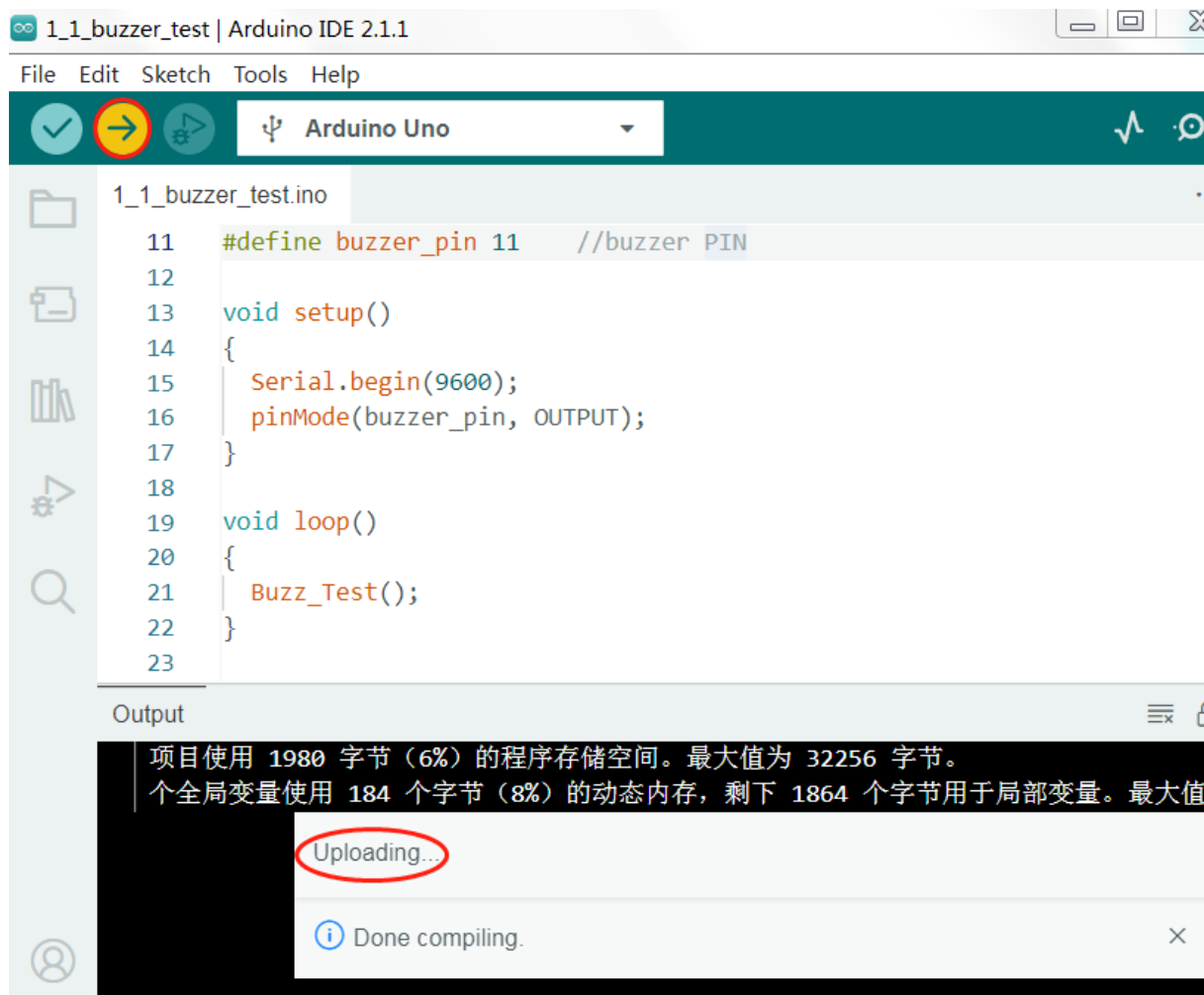
Click compile button , successfully compiled the code will display “Done compiling”



5.6 Upload

Before uploading the code, turn the ESP-01 switch on the control board to the side away from the "ESP-01" silk screen.

Click upload button , successfully uploading the code will display “Done uploading”. After the code is uploaded successfully, the program starts to run automatically, and you can hear the beeping sound from the buzzer on the control board



5.7 Code

Buzzer_test code:

```
#define buzzer_pin 11    //buzzer PIN is 11 on the arduino 4WD shield

void setup()
{
    Serial.begin(9600); //Set baud rate
    pinMode(buzzer_pin, OUTPUT); //Set pin 11 to output mode and input signals to the
    buzzer
}

void loop()
{
    for(int i = 0; i < 80; i++)
    {
        //The high and low levels constitute a square-wave signal that triggers the passive
        buzzer to work
        digitalWrite(buzzer_pin, HIGH);
        delay(1);
        digitalWrite(buzzer_pin, LOW);
        delay(1);
    }
    for(int j = 0; j < 100; j++)
    {
        digitalWrite(buzzer_pin, HIGH);
        delay(2);
        digitalWrite(buzzer_pin, LOW);
        delay(2);
    }
}
```


6 Extended experiment

Let's try making the buzzer play the melody of a song next.

6.1 Open the buzzer_sing code with Arduino IDE, you can find the code in this folder:

E:\CKK0002-master\Tutorial\sketches\1_2_buzzer_sing

6.2 Compile and upload the code to the control board, after the upload is successful, the program runs, and you will hear the buzzer start singing.

6.3 Code

Buzzer_sing code:

```
//Tenor NTF 0 is an empty beat
#define NTF0 -1
#define NTF1 350
#define NTF2 393
#define NTF3 441
#define NTF4 495
#define NTF5 556
#define NTF6 624
#define NTF7 661

//High pitch NTFH
#define NTFH1 700
#define NTFH2 786
#define NTFH3 882
#define NTFH4 935
#define NTFH5 965
#define NTFH6 996
#define NTFH7 1023

//Low pitch NTFL
#define NTFL1 175
#define NTFL2 196
#define NTFL3 221
#define NTFL4 234
#define NTFL5 262
#define NTFL6 294
#define NTFL7 330
//Note frequency array
int tune[]=
{
  NTF3,NTF3,NTF3,NTF3,NTF3,NTF3,
  NTF3,NTF5,NTF1,NTF2,NTF3,NTF0,
  NTF4,NTF4,NTF4,NTF4,NTF4,NTF3,NTF3,NTF3,
```

```
NTF5,NTF5,NTF4,NTF2,NTF1,NTF0,

NTFL5,NTF3,NTF2,NTF1,NTFL5,NTF0,NTFL5,NTFL5,
NTFL5,NTF3,NTF2,NTF1,NTFL6,NTF0,
NTFL6,NTF4,NTF3,NTF2,NTFL7,NTF0,
NTF5,NTF5,NTF4,NTF2,NTF3,NTF1,NTF0,

NTFL5,NTF3,NTF2,NTF1,NTFL5,NTF0,
NTFL5,NTF3,NTF2,NTF1,NTFL6,NTF0,NTFL6,
NTFL6,NTF4,NTF3,NTF2,NTF5,NTF5,NTF5,NTF5,
NTF6,NTF5,NTF4,NTF2,NTF1,NTF0
};
//Note beat array
float durt[]=
{
    0.5,0.5,1,0.5,0.5,1,
    0.5,0.5,0.75,0.25,1.5,0.5,
    0.5,0.5,1,0.5,0.5,0.5,0.5,0.25,0.25,
    0.5,0.5,0.5,0.5,1.5,0.5,

    0.5,0.5,0.5,0.5,1,0.5,0.25,0.25,
    0.5,0.5,0.5,0.5,1,1,
    0.5,0.5,0.5,0.5,1,1,
    0.5,0.5,0.5,0.5,1,0.75,0.25,

    0.5,0.5,0.5,0.5,1,1,
    0.5,0.5,0.5,0.5,1,0.5,0.5,
    0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,
    0.5,0.5,0.5,0.5,0.75,0.25
};
//Define the buzzer pin, note length variable
int buzzer_pin = 11;
int length;

//initializes the pin and calculates the length
void setup()
{
    pinMode(buzzer_pin, OUTPUT);
    length = sizeof(tune)/sizeof(tune[0]);
}

//loop
void loop()
{
    //for looping
    for(int x=0;x<length;x++)
    {
        tone(buzzer_pin, tune[x]);
        delay(500*durt[x]); //The 500 here controls the length of each note to
determine the rhythm of the piece
        noTone(buzzer_pin);
    }
    delay(500); //The interval between starting the next cycle
```

}

7 Any questions and suggestions are welcome

Thank you for reading this document!

If you find any errors and omissions in the tutorial, or if you have any suggestions and questions, please feel free to contact us:
cokoino@outlook.com

We will do our best to make changes and publish revisions as soon as possible.

If you want to learn more about smart cars, robots, learning kits and other technology products from us, please bookmark and pay attention to our website:

<http://cokoino.com/>

We will continue to launch interesting, cost-effective, innovative, user-friendly products.

LK COKOINO