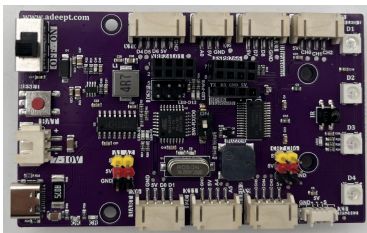



Lesson 6 The Buzzer Plays Music

6.1 Overview

This lesson focuses on using the buzzer on the Adeept Robot Control Board to play various tones and a song. By following the steps and code examples provided, you can learn how to control the buzzer's frequency and duration to create different musical sounds.

6.2 Required Components

Components	Quantity	Picture
Adeept Robot Control Board	1	
Type-C USB Cable	1	

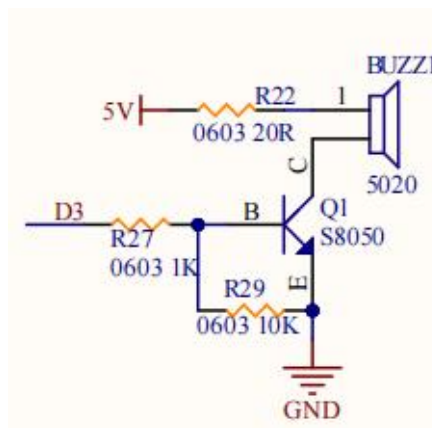
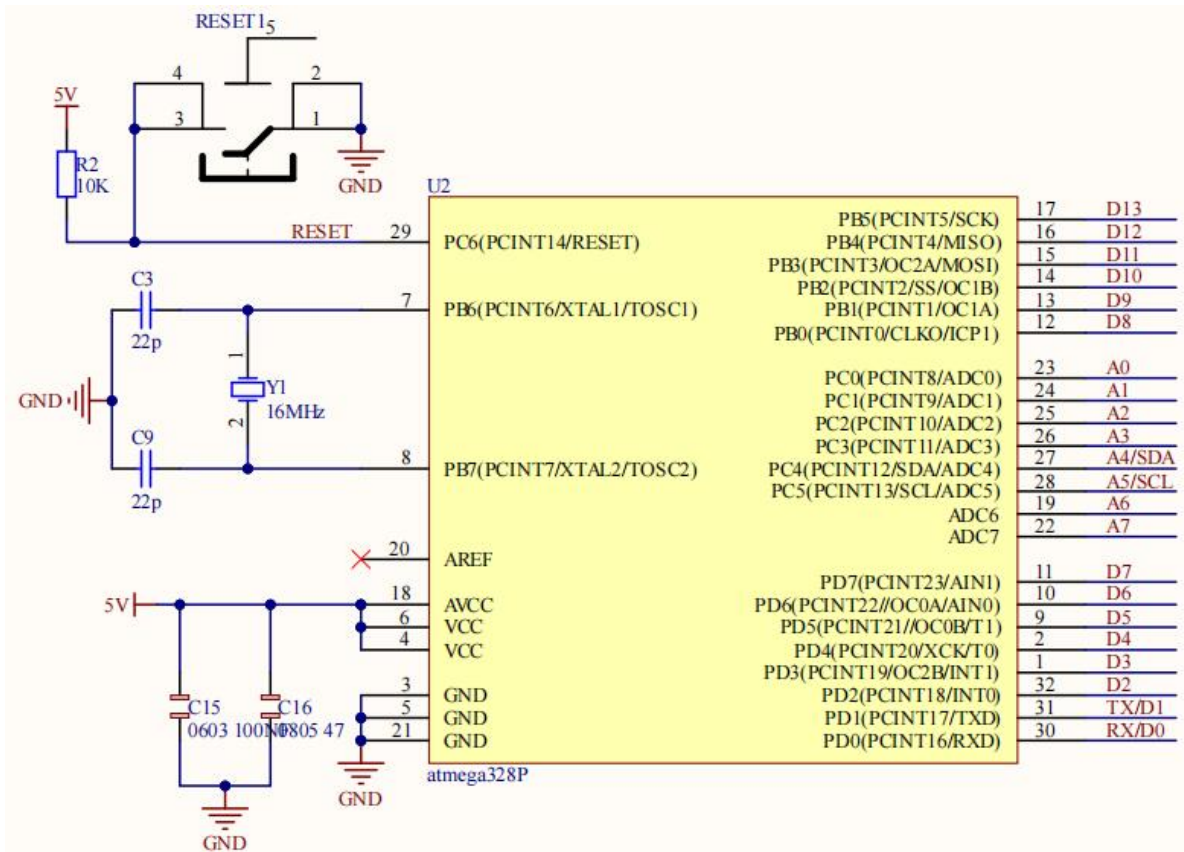
6.3 Principle Introduction

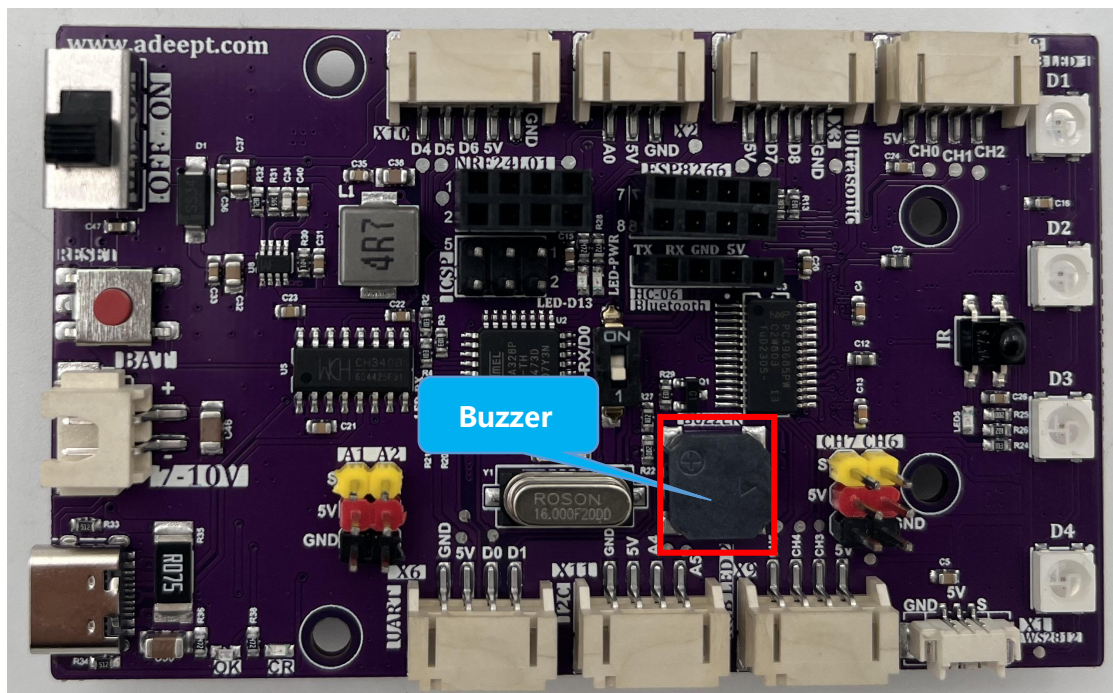
The Adeept Robot Control Board has a passive buzzer integrated into it. The buzzer is connected to pin D3 of the board. When an appropriate electrical signal is sent to this pin, the buzzer can produce sounds of different frequencies. The frequency of the electrical signal determines the pitch of the sound, and the duration of the signal controls how long the sound lasts.

PINS of Adeept Robot Control Board	Buzzer
D3	Buzzer

6.4 Wiring Diagram

A passive buzzer is built into the Adeept Robot Control board.



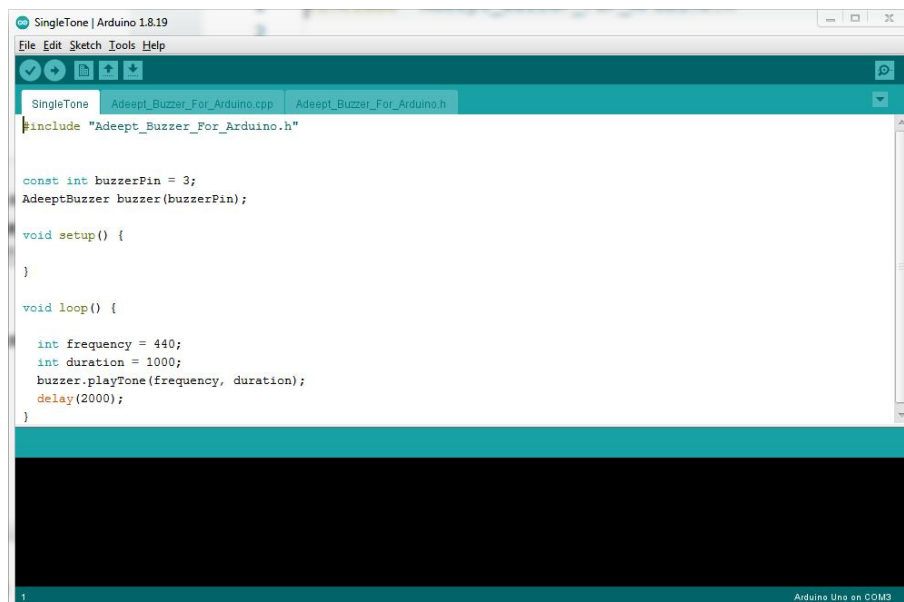


6.5 Demonstration

1. Connect your computer and Adept Robot Control Board (Arduino Board) with a USB cable.

Play a tone

2. Open "02_Buzzer/SingleTone" folder in "/Code/" , double-click "SingleTone.ino" .

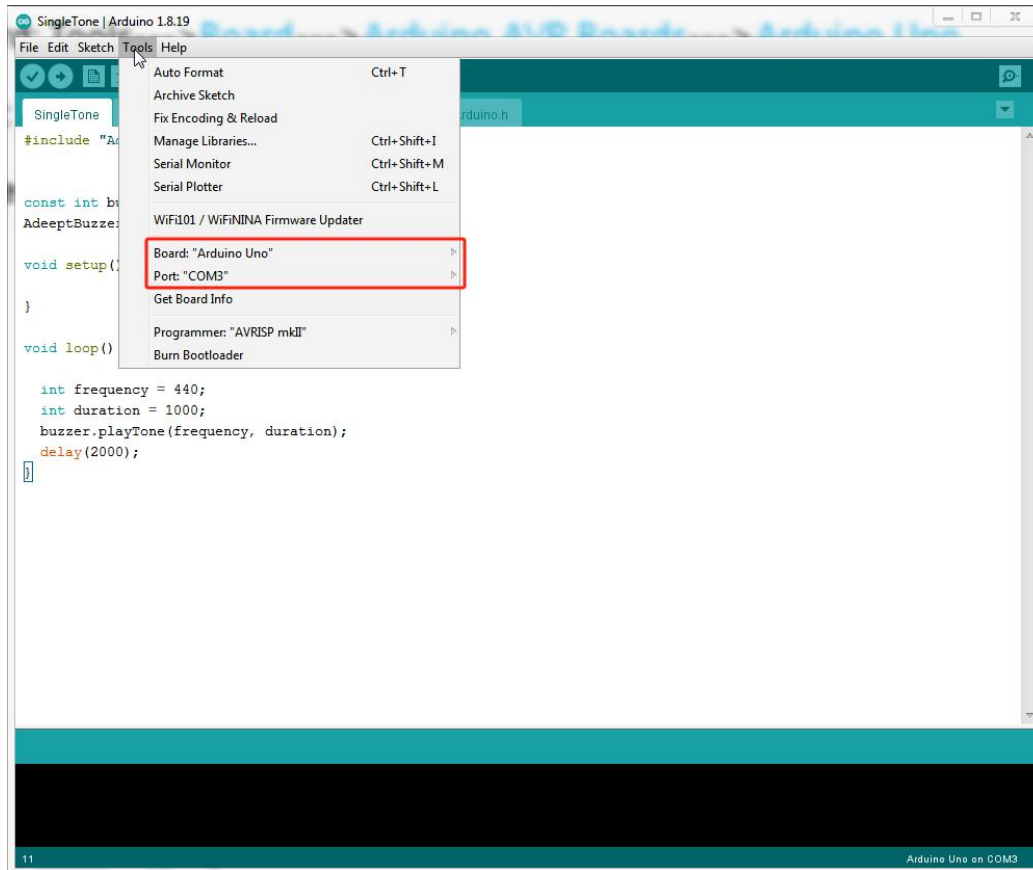



3. Select development board and serial port.

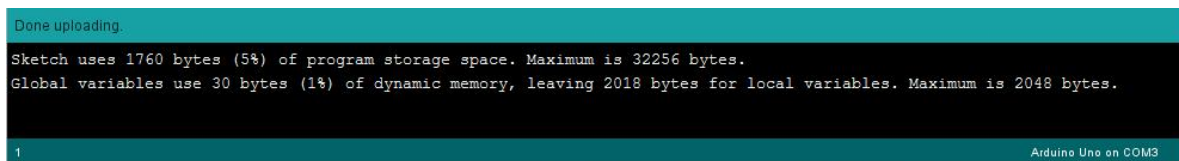
Board: Tools--->Board--->Arduino AVR Boards--->Arduino Uno

Port: Tools --->Port--->COMx

Note: The port number will be different in different computers.



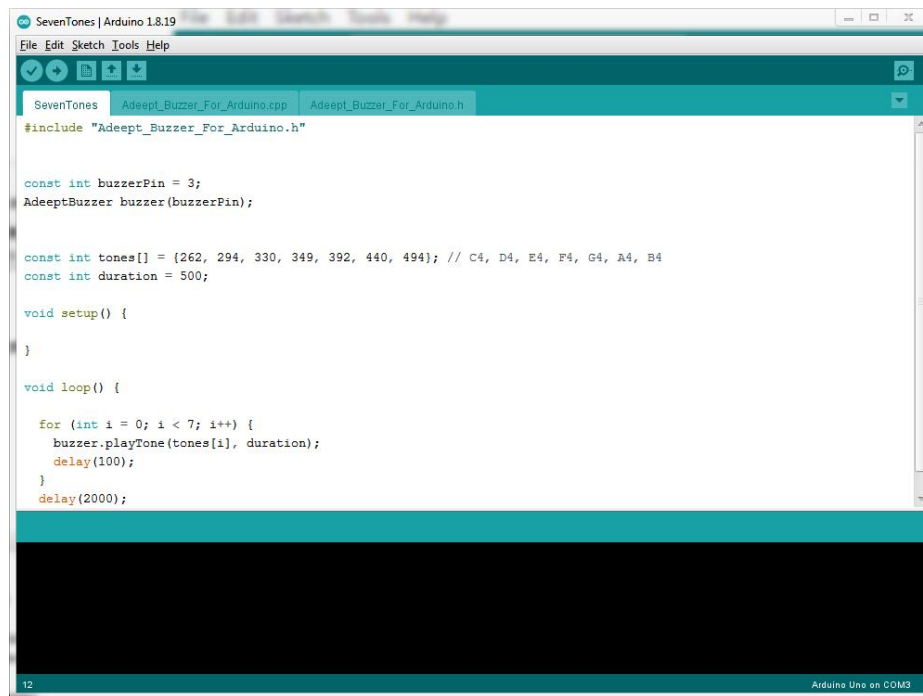
4. After opening, click  to upload the code program to the Arduino. If there is no error warning in the console below, it means that the Upload is successful.



5. After the program runs successfully, the onboard buzzer will emit a tone every 1 second, with an interval of 1 second after each sound, and so on.

Play 7 tones

6. Open "02_Buzzer/SevenTones" folder in "/Code/" , double-click "SevenTones.ino" .



```
SevenTones | Arduino 1.8.19
File Edit Sketch Tools Help

SevenTones Adeept_Buzzer_For_Arduino.cpp Adeept_Buzzer_For_Arduino.h
#include "Adeept_Buzzer_For_Arduino.h"


const int buzzerPin = 3;
AdeeptBuzzer buzzer(buzzerPin);

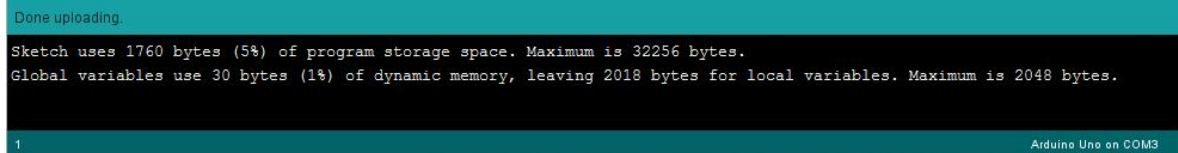
const int tones[] = {262, 294, 330, 349, 392, 440, 494}; // C4, D4, E4, F4, G4, A4, B4
const int duration = 500;

void setup() {
}

void loop() {
  for (int i = 0; i < 7; i++) {
    buzzer.playTone(tones[i], duration);
    delay(100);
  }
  delay(2000);
}
```



7. After opening, click  to upload the code program to the Arduino. If there is no error warning in the console below, it means that the Upload is successful.

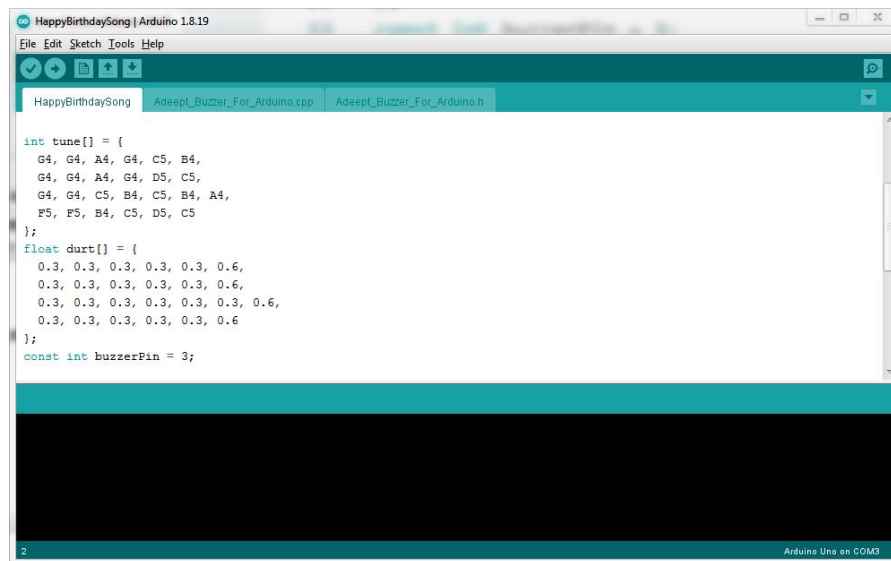



```
Done uploading.
Sketch uses 1760 bytes (5%) of program storage space. Maximum is 32256 bytes.
Global variables use 30 bytes (1%) of dynamic memory, leaving 2018 bytes for local variables. Maximum is 2048 bytes.
```

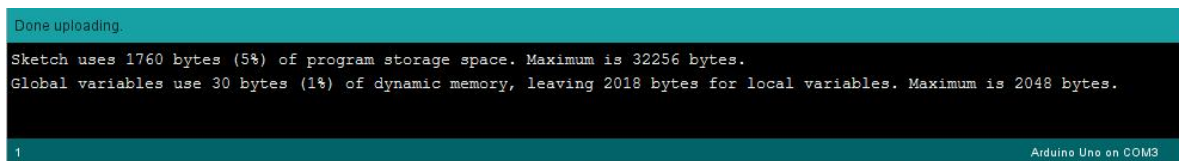
8. After the program runs successfully, the onboard buzzer will play the tones of the seven notes C4, D4, E4, F4, G4, A4, and B4 in sequence, with each tone lasting 500 milliseconds and an interval of 100 milliseconds between them. After completing one round of playing the seven notes, it will pause for 2 seconds and then repeat the playback.

Play Happy Birthday song

9. Open **"02_Buzzer/HappyBirthdaySong"** folder in **"/Code/"** , double-click **"HappyBirthdaySong.ino"** .



10. After opening, click  to upload the code program to the Arduino. If there is no error warning in the console below, it means that the Upload is successful.



11. After the program runs successfully, the onboard buzzer will play the "Happy Birthday" song.

6.6 Code

Complete code refer to [SingleTone.ino](#)

```

01 #include "Adeept_Buzzer_For_Arduino.h"
02
03
04 const int buzzerPin = 3;
05 AdeeptBuzzer buzzer(buzzerPin);
06
07 void setup() {
08
09 }
10
11 void loop() {
12
13     int frequency = 440;
14     int duration = 1000;
15     buzzer.playTone(frequency, duration);
16     delay(2000);
17 }

```

Complete code refer to [SevenTones.ino](#)

```
01 #include "Adeept_Buzzer_For_Arduino.h"
02
03
04 const int buzzerPin = 3;
05 AdeeptBuzzer buzzer(buzzerPin);
06
07
08 const int tones[] = {262, 294, 330, 349, 392, 440, 494}; // C4, D4, E4, F4, G4, A4, B4
09 const int duration = 500;
10
11 void setup() {
12
13 }
14
15 void loop() {
16
17     for (int i = 0; i < 7; i++) {
18         buzzer.playTone(tones[i], duration);
19         delay(100);
20     }
21     delay(2000);
22 }
```

Complete code refer to [HappyBirthdaySong.ino](#)

```
01 #include "Adeept_Buzzer_For_Arduino.h"
02
03 #define G4 392
04 #define A4 440
05 #define B4 494
06 #define C5 523
07 #define D5 587
08 #define F5 698
09
10
11 int tune[] = {
12     G4, G4, A4, G4, C5, B4,
13     G4, G4, A4, G4, D5, C5,
14     G4, G4, C5, B4, C5, B4, A4,
15     F5, F5, B4, C5, D5, C5
16 };
17 float durt[] = {
18     0.3, 0.3, 0.3, 0.3, 0.3, 0.6,
19     0.3, 0.3, 0.3, 0.3, 0.3, 0.6,
20     0.3, 0.3, 0.3, 0.3, 0.3, 0.6,
21     0.3, 0.3, 0.3, 0.3, 0.3, 0.6
22 };
23 const int buzzerPin = 3;
24
25 AdeeptBuzzer buzzer(buzzerPin);
26 int length = sizeof(tune) / sizeof(tune[0]);
27 void setup() {
28
29 }
```



```

30
31 void loop() {
32
33     for (int i = 0; i < length; i++) {
34         int duration = durt[i] * 1000;
35         buzzer.playTone(tune[i], duration);
36         delay(10);
37     }
38     delay(2000);
39 }

```

Complete code refer to [Adeept_Buzzer_For_Arduino.cpp](#)

```

01 #include "Adeept_Buzzer_For_Arduino.h"
02 #include <Arduino.h>
03
04 AdeeptBuzzer::AdeeptBuzzer(int pin) {
05     buzzerPin = pin;
06     pinMode(buzzerPin, OUTPUT);
07 }
08
09 void AdeeptBuzzer::playTone(int frequency, int duration) {
10     tone(buzzerPin, frequency, duration);
11     delay(duration);+
12     noTone(buzzerPin);
13 }
14

```

Complete code refer to [Adeept_Buzzer_For_Arduino.h](#)

```

01 // Adeept_Buzzer_For_Arduino.h
02 #ifndef ADEEPT_BUZZER_FOR_ARDUINO_H
03 #define ADEEPT_BUZZER_FOR_ARDUINO_H
04
05 class AdeeptBuzzer {
06     public:
07         AdeeptBuzzer(int pin);
08         void playTone(int frequency, int duration);
09     private:
10         int buzzerPin;
11 };
12
13 #endif

```

Code explanation

SingleTone.ino

Initialization phase:

Call the setup () function to complete hardware initialization. In the current code, the interior of the setup() function is temporarily empty and no specific initialization operation has been performed.

Loop control process:

Stage 1: Drive the car buzzer to sound at a frequency of 440Hz → Delay for 1 second

Stage 2: After the buzzer stops sounding → Delay for 2 second

Forming a cyclic effect, Make the buzzer periodically emit a sound that lasts for 1 second and has a frequency of 440Hz.

SevenTones.ino

Initialization phase:

Call the setup () function to complete hardware initialization. In the current code, the interior of the setup() function is temporarily empty and no specific initialization operation has been performed.

Loop control process:

Stage 1: Drive the onboard buzzer to sound at frequencies of 262Hz, 294Hz, 330Hz, 349Hz, 392Hz, 440Hz, and 494Hz (corresponding to notes C4, D4, E4, F4, G4, A4, B4) in sequence, with each frequency lasting 500 milliseconds and a delay of 100 milliseconds after each tone is played.

Stage 2: After completing a round of playing seven different frequency tones, the buzzer stops sounding and the program enters a delay state, waiting for 2 seconds.

Forming a cyclic effect, The buzzer will periodically play these seven different frequency tones in sequence, each lasting 500 milliseconds with a 100 millisecond interval between them, and pause for 2 seconds after completing each round of playback.

HappyBirthdaySong.ino

Initialization phase:

Call the setup () function to complete hardware initialization. In the current code, the interior of the setup() function is temporarily empty and no specific initialization operation has been performed.

Loop control process:

Stage 1: Drive the onboard buzzer to sound sequentially based on the preset note frequency array tune and duration array dur. Note frequency includes G4 (392Hz) 、 A4 (440Hz) 、 B4 (494Hz) 、 C5 (523Hz) 、 D5 (587Hz) 、 F5 (698Hz) Wait, the duration of each note is determined by multiplying the corresponding value in the dur array by 1000 (in milliseconds). After playing each note, the program will pause for 10 milliseconds.

Stage 2: After completing a round of playing all preset notes, the buzzer stops sounding and the program enters a delay state, waiting for 2 seconds.

Forming a cyclic effect, Make the buzzer play periodically according to the preset note sequence and duration, and pause for 2 seconds after completing each round of playback.