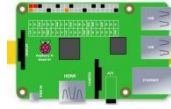



Lesson 8 Use the Buzzer to Play Music

8.1 Overview

This lesson will explore using Raspberry Pi and Adeept Robot HAT V3.2 to make the buzzer play music. First, understand the components, principles, and wiring, then run a Python program to demonstrate playing different notes and songs.

8.2 Required Components

Components	Quantity	Picture
Raspberry Pi	1	
Adeept Robot HAT V3.2	1	

8.3 Principle Introduction

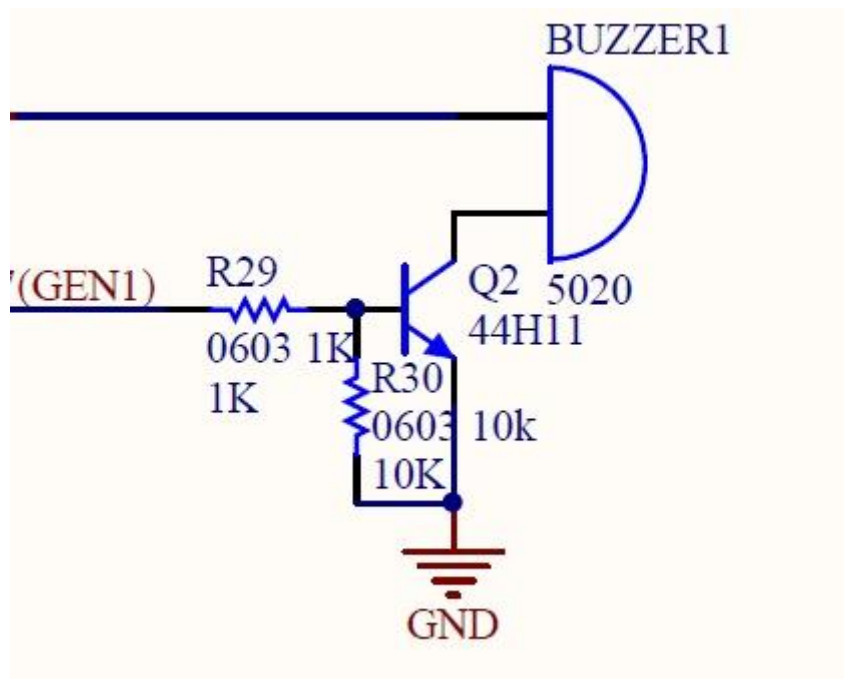
A buzzer is an electronic device that can produce sound. The onboard buzzer on Adeept Robot HAT V3.2 operates based on electromagnetic principles. When current passes through the coil inside the buzzer, a magnetic field is generated. This magnetic field interacts with the internal components of the buzzer, causing the diaphragm or vibrating element to move. As the diaphragm vibrates, it generates sound waves in the surrounding air.

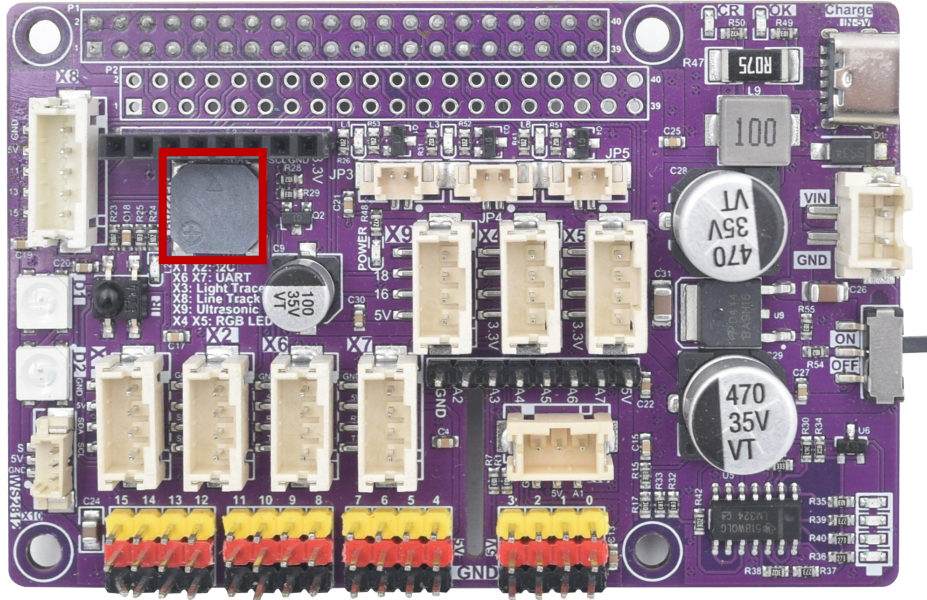
In the environment of Raspberry Pi, we use GPIO (General Purpose Input/Output) pins to control the buzzer. The GPIO18 pin connected to the buzzer can output different electrical signals through programming. By sending a series of electrical pulses with specific frequency and duration to the buzzer through the GPIO18 pin, we can make it emit different notes. For example, higher frequency electrical signals produce higher pitch notes, while lower frequency signals produce lower pitch notes. The duration of each signal determines the playback time of the notes. We use the Python programming language to generate these electrical signals by controlling GPIO pins, thereby creating melodies and playing music.

PINS of Raspberry Pi	Sensor
GPIO18	Buzzer

8.4 Wiring Diagram

The motherboard of Adeept Robot HAT V3.2 is equipped with an onboard buzzer. This onboard buzzer is connected to the GPIO18 pin.





8.5 Demonstration

Run the code

1. **Remotely log:** Remotely log in to the Raspberry Pi terminal..
2. **Navigate to the Program Folder:** Enter the following command in the terminal and press Enter to access the folder where the program is located:

```
cd Adeept_PiCar-Pro/Examples/02_Buzzer/
```

```
pi@raspberrypi:~ $ cd Adeept_PiCar-Pro/Examples/02_Buzzer/  
pi@raspberrypi:~/Adeept PiCar-Pro/Examples/02_Buzzer $
```

3. **View Directory Contents:** Type "ls" in the terminal and press Enter. This will display all the files in the current directory, ensuring that the "**SingleTone.py**", "**SevenNotes.py**" and "**HappyBirthday.py**" file is present:

```
ls
```

```
pi@raspberrypi:~/Aadept_PiCar-Pro/Examples/02_Buzzer $ ls
HappyBirthday.py  SevenNotes.py  SingleTone.py
```

4. **Run the Program:**Demonstrate how to use a buzzer to play a single note.

```
sudo python3 SingleTone.py
```

```
pi@raspberrypi:~/Aadept_PiCar-Pro/Examples/02_Buzzer $ sudo python3 SingleTone.py
Demo: Playing a single note
C4
```

5. **Observation and Termination:**After successfully running the program, the buzzer will sound a note. When you want to terminate the running program, you can press the "**Ctrl + C**" shortcut key on the keyboard.

6. **Run the Program:**Demonstrate how to use a buzzer to play seven basic notes.

```
sudo python3 SevenNotes.py
```

```
pi@raspberrypi:~/Aadept_PiCar-Pro/Examples/02_Buzzer $ sudo python3 SevenNotes.py
Demo: Introducing 7 musical notes
C4
D4
E4
F4
G4
A4
B4
```

7. **Observation and Termination:**After successfully running the program, you will hear the buzzer emit seven basic notes. When you want to terminate the running program, you can press the "**Ctrl+C**" shortcut key on the keyboard.

8. **Run the Program:**Demonstrate how to use a buzzer to play the song "Happy Birthday" .

```
sudo python3 HappyBirthday.py
```

```
pi@raspberrypi:~/Adeept_PiCar-Pro/Examples/02_Buzzer $ sudo python3 HappyBirthday.py
Demo: Playing the Happy Birthday song
G4
G4
A4
G4
C5
B4
G4
G4
```

9. Observation and Termination:After successfully running the program, the buzzer will play the song "Happy Birthday"., When you want to terminate a running program, you can press the **Ctrl+C** shortcut key on the keyboard.

8.6 Code

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

```
01  #!/usr/bin/env/python
02  # File name   : SingleTone.py
03  # Website    : www.Adeept.com
04  # Author     : Adeept
05  # Date      : 2025/03/5
06  from gpiozero import TonalBuzzer
07  from time import sleep
08
09  # Initialize a TonalBuzzer connected to GPIO18 (BCM)
10  tb = TonalBuzzer(18)
11
12  # Define a single note
13  SINGLE_NOTE = [("C4", 0.5)]
14
15  def play(tune):
16      """
17      Play a musical tune using the buzzer.
18      :param tune: List of tuples (note, duration),
19      where each tuple represents a note and its duration.
20      """
21      for note, duration in tune:
22          print(note) # Output the current note being played
23          tb.play(note) # Play the note on the buzzer
24          sleep(float(duration)) # Delay for the duration of the note
25          tb.stop() # Stop playing after the tune is complete
26
27  if __name__ == "__main__":
28      try:
29          # First demo: Play a single note
```

```

30     print("Demo: Playing a single note")
31     play(SINGLE_NOTE)
32
33     except KeyboardInterrupt:
34         # Handle KeyboardInterrupt for graceful termination
35         tb.stop()
36         print("Program terminated by user.")

```

Complete code refer to [SevenNotes.py](#).

```

01     01  #!/usr/bin/env/python
02     02  # File name    : SevenNotes.py
03     03  # Website     : www.Adeept.com
04     04  # Author      : Adeept
05     05  # Date       : 2025/03/5
06     06  from gpiozero import TonalBuzzer
07     07  from time import sleep
08     08
09     09  # Initialize a TonalBuzzer connected to GPIO18 (BCM)
10     10  tb = TonalBuzzer(18)
11     11
12     12  # Define 7 musical notes, the following are the frequencies corresponding to the seven musical
13     13  notes.
14     14  #C4-261.63 D4-293.66 E4-328.63 F4-348.23 G4-392.00 A4-440.00 B4-
15     15  493.88
16     16  SEVEN_NOTES = [
17     17      ["C4", 0.5], ["D4", 0.5], ["E4", 0.5], ["F4", 0.5],
18     18      ["G4", 0.5], ["A4", 0.5], ["B4", 0.5]
19     19  ]
20     20
21     21
22     22  def play(tune):
23     23      """
24     24      Play a musical tune using the buzzer.
25     25      :param tune: List of tuples (note, duration),
26     26      where each tuple represents a note and its duration.
27     27      """
28     28      for note, duration in tune:
29     29          print(note) # Output the current note being played
30     30          tb.play(note) # Play the note on the buzzer
31     31          sleep(float(duration)) # Delay for the duration of the note
32     32          tb.stop() # Stop playing after the tune is complete
33     33
34     34  if __name__ == "__main__":
35     35      try:
36     36          # Second demo: Introduce 7 musical notes
37     37          print("Demo: Introducing 7 musical notes")
38     38          play(SEVEN_NOTES)
39     39      except KeyboardInterrupt:
40     40          # Handle KeyboardInterrupt for graceful termination
          tb.stop()
          print("Program terminated by user.")

```

Complete code refer to [HappyBirthday.py](#).

```
01  #!/usr/bin/env/python
02  # File name   : HappyBirthday.py
03  # Website    : www.Adeept.com
04  # Author     : Adeept
05  # Date      : 2025/03/5
06  from gpiozero import TonalBuzzer
07  from time import sleep
08
09  # Initialize a TonalBuzzer connected to GPIO18 (BCM)
10  tb = TonalBuzzer(18)
11
12  # Define the "Happy Birthday" song
13  HAPPY_BIRTHDAY_SONG = [
14      ["G4", 0.3], ["G4", 0.3], ["A4", 0.3], ["G4", 0.3], ["C5", 0.3], ["B4", 0.6],
15      ["G4", 0.3], ["G4", 0.3], ["A4", 0.3], ["G4", 0.3], ["D5", 0.3], ["C5", 0.6],
16      ["G4", 0.3], ["G4", 0.3], ["C5", 0.3], ["B4", 0.3], ["C5", 0.3], ["B4", 0.3], ["A4", 0.6],
17      ["F5", 0.3], ["F5", 0.3], ["B4", 0.3], ["C5", 0.3], ["D5", 0.3], ["C5", 0.6]
18  ]
19
20  def play(tune):
21      """
22      Play a musical tune using the buzzer.
23      :param tune: List of tuples (note, duration),
24      where each tuple represents a note and its duration.
25      """
26      for note, duration in tune:
27          print(note) # Output the current note being played
28          tb.play(note) # Play the note on the buzzer
29          sleep(float(duration)) # Delay for the duration of the note
30          tb.stop() # Stop playing after the tune is complete
31
32  if __name__ == "__main__":
33      try:
34          # Third demo: Play the entire "Happy Birthday" song
35          print("Demo: Playing the Happy Birthday song")
36          play(HAPPY_BIRTHDAY_SONG)
37
38      except KeyboardInterrupt:
39          # Handle KeyboardInterrupt for graceful termination
40          tb.stop()
41          print("Program terminated by user.")
```

Code explanation

[SingleTone.py](#)

Initialization Stage:

Initialize the buzzer connected to GPIO18 by calling `TonalBuzzer(18)`.

Define the tone parameters: `SINGLE_NOTE = [["C4", 0.5]]` (Middle C, with a duration of 0.5 seconds).

Playback Process

Enter the single - play process and execute the following steps sequentially:

Play the C4 tone → Last for 0.5 seconds → Stop automatically.

Print the name of the currently playing note in real - time.

When Ctrl+C is detected:

Stop the buzzer immediately, Output an exit prompt, Terminate the program.

SevenNotes.py

Initialization Stage:

Initialize the buzzer connected to GPIO18 using `TonalBuzzer`.

Scale Definition:

Pre - Define 7 musical notes, the following are the frequencies corresponding to the seven musical notes.

```
C4-261.63 D4-293.66 E4-328.63 F4-348.23 G4-392.00 A4-440.00 B4-493.88
```

Playback Process:

Play the seven notes in sequence (from C4 to B4).

Print the note name (such as "C4") when each note is played.

Automatically stop after all notes have been played.

Press Ctrl+C to immediately mute the buzzer and exit the program.

HappyBirthday.py

Initialization Stage:

Initialize the buzzer connected to GPIO18 using `TonalBuzzer`.

Song Definition:

Pre - define the complete melody of "Happy Birthday", including notes from G4 to F5 (for example, ["G4", 0.3] means the G4 note lasts for 0.3 seconds).

Playback Process:

Play all 24 notes in sequence to form the complete song.

Print the note name (such as "C5") when each note is played.

Alternate between short notes (0.3 seconds) and long notes (0.6 seconds) to simulate the rhythm of the song.

Press Ctrl+C to immediately mute the buzzer and exit the program.

The complete playback takes about 9 seconds, realizing the electronic buzzer performance of the classic "Happy Birthday" melody.