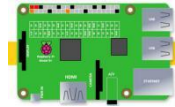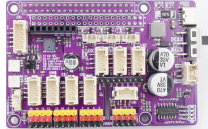# Lesson 6 How to Control Buzzer

In this lesson, we will learn how to control Buzzer.

6.1 Components used in this course

Components	Quantity	Picture
Raspberry Pi	1	
Adeept Robot HAT V3.0	1	

5.2 Introduction of Buzzer

What is a Buzzer?

The buzzer is an integrated electronic alarm device powered by DC power supply. It is widely used in the sound production of computers, printers, copiers, alarms, electronic toys, vehicle electronic equipment, telephones, timers and other equipment.

Buzzers are divided into active buzzers and passive buzzers.

There is no distinction between positive and negative for a passive buzzer. Similar to a speaker, the sound can be achieved by loading electrical signals of different frequencies on the two pins. The sounds emitted are different according to different frequencies.

The active buzzer is divided into positive and negative. It only needs to add a voltage signal to the two legs to make a sound. The sound produced has a single tone and a fixed frequency.

Active buzzers have more oscillating structures than passive buzzers.

Adeept Robot HAT V3.0 has a passive buzzer onboard that can emit sounds of different frequencies. The onboard buzzer can be controlled through the GPIO18 (BCM) pin.
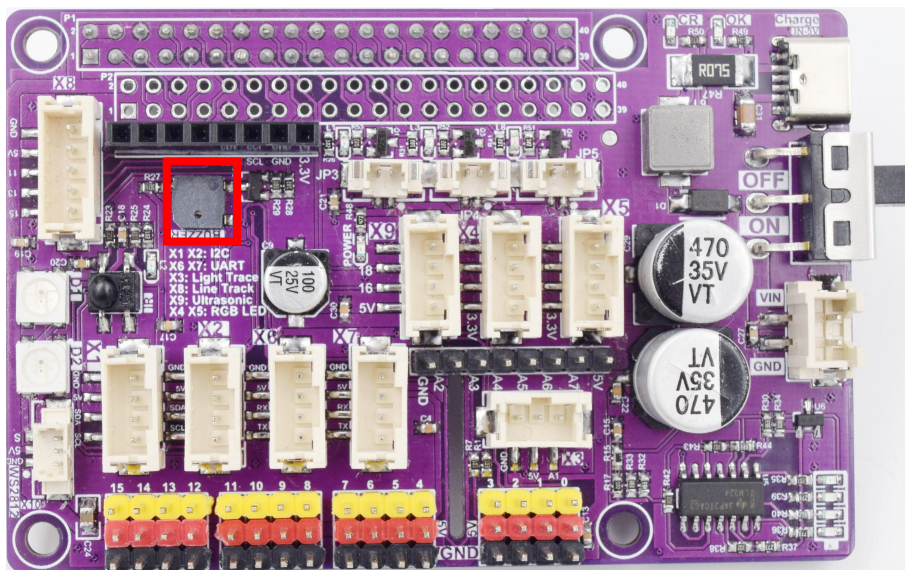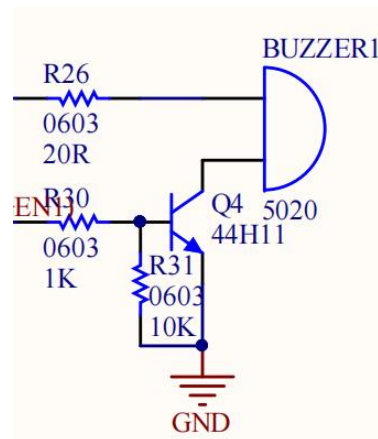
6.2 How to control Buzzer

Run the code

1. Remotely log in to the Raspberry Pi terminal.

```
Linux raspberrypi 4.19.118-v7l+ #1311 SMP Mon Apr 27 14:26:42 BST 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Aug 29 08:17:49 2020 from 192.168.3.208

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
 a new password.

pi@raspberrypi:~ $
```

2. Enter the command and press Enter to enter the folder where the program is located:

cd adeept_rasptank2/examples/

```
pi@raspberrypi:~ $
pi@raspberrypi:~ $ cd adeept_rasptank2/examples/
pi@raspberrypi:~/adeept_rasptank2/examples $
pi@raspberrypi:~/adeept_rasptank2/examples $
```

3. View the contents of the current directory file:

ls

```
pi@raspberrypi:~/adeept_rasptank2/examples $ ls
01_LED.py     03_servo.py   05_ws2812.py   07_lineTracking.py
02_buzzer.py  04_motor.py   06_ultra.py
pi@raspberrypi:~/adeept_rasptank2/examples $
```

4. Enter the command and press Enter to run the program:

Older versions of the Raspberry Pi operating system include Python2 and python3.

In order for the program to run properly in older versions of Raspberry Pi OS, use "python3" to run the program. Of course, in new versions of Raspberry Pi OS, you can also use the "python" command to run programs.

In order to prevent users from misunderstanding, this textbook uses the "**python3**" command to run the program.

sudo python3 02_buzzer.py

5. After running the program successfully, You will hear the buzzer sounding the tone of the song. The program will terminate after a few seconds..

6. When you want to terminate the running program, you can press the shortcut key "Ctrl + C" on the keyboard.

6.3 The main code program

Complete code refer to 02_buzzer.py.

```
1.  #!/usr/bin/env python3
2.  from gpiozero import TonalBuzzer
3.  from time import sleep
4.
5.  # Initialize a TonalBuzzer connected to GPIO18 (BCM)
6.  tb = TonalBuzzer(18)
7.
8.  # Define a musical tune as a sequence of notes and durations.
9.  SONG = [
```

```
10.    ["E5",0.3],["Eb5",0.3],
11.    ["E5",0.3],["Eb5",0.3],["E5",0.3],["B4",0.3],["D5",0.3],["C5",0.3],
12.    ["A4",0.6],[None,0.1],["C4",0.3],["E4",0.3],["A4",0.3],
13.    ["B4",0.6],[None,0.1],["E4",0.3],["Ab4",0.3],["B4",0.3],
14.    ["C5",0.6],[None,0.1],["E4",0.3],["E5",0.3],["Eb5",0.3],
15.    ["E5",0.3],["Eb5",0.3],["E5",0.3],["B4",0.3],["D5",0.3],["C5",0.3],
16.    ["A4",0.6],[None,0.1],["C4",0.3],["E4",0.3],["A4",0.3],
17.    ["B4",0.6],[None,0.1],["E4",0.3],["C5",0.3],["B4",0.3],["A4",0.1]
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.         play(SONG)  # Execute the play function to start playing the tune.
35.
36.     except KeyboardInterrupt:
37.         # Handle KeyboardInterrupt for graceful termination
38.         pass
```