1.2.3 Voice Wake-Up

In this lesson, we'll learn how to use a large speech model to activate the voice device by speaking a predefined wake word through a program.

1. WonderEcho Pro Wake-Up

1.1. Device Check

To proceed, we need to identify the USB device name assigned to the connected WonderEcho Pro or Circular Microphone Array (hereafter referred to as the voice device). Please follow the steps below carefully. (**Note:** Do not connect any other USB devices during this process to avoid confusion.)

- First, disconnect the voice device, then open a terminal and run the following command:

```
ll /dev | grep USB
```

- Next, reconnect the voice device to the USB port on your main board and run the same command again:

```
ll /dev | grep USB
```

- You should now see a newly listed USB port, such as ttyCH341USB1. Please take note of this device name—it may vary depending on the main controller being used.

1.2. Wake-Up Test

1) To begin, update the port number used in the program by editing the script. You'll also need to uncomment the line for the port you're using and comment out any unused ports.

```
vim  wakeup_demo.py
```

Press i to enter edit mode and make the necessary changes as shown below (update the port number accordingly and adjust comments as needed).

```
port = '/dev/ttyCH341USB1'
kws = awake.WonderEchoPro(port)
# kws = awake.CircleMic(port, 'hello hi wonder', 'mic6_circle', True)
```

Once the changes are complete, press ESC, then type :wq and press Enter to save and exit the editor.

2) Next, return to the system interface and run the wake-up demo using the command below. Speak the wake word "HELLO HIWONDER" clearly toward the WonderEcho Pro voice device.
If the output includes "keyword detect", it indicates that the firmware has been successfully flashed and the wake word is functioning correctly.

```
python3  ~/large_models/wakeup_demo.py
```

```
www.hiwonder.com
sh: 1: pinctrl: not found
start...
hello hiwonder
```

2. 6-Microphone Circular Array

As with the WonderEcho Pro, you can connect the 6-Microphone Circular

Array to your main board (Raspberry Pi or NVIDIA Jetson) using a Type-C to USB cable.

2.1. Device Check

For Jetson users, connect to the Jetson system using the NoMachine remote access tool. Once connected, check the desktop interface.
If the 6-Microphone Circular Array icon appears on the left side of the desktop, it indicates the device has been successfully recognized.

2.2. Wake-Up Test

1) Open a new terminal window and run the following command to edit the wakeup_demo.py script:

```
vim  ~/large_models/wakeup_demo.py
```

2) Press i to enter edit mode.

3) Update the port to match the device port number you previously identified. Comment out the WonderEcho Pro configuration (add # at the beginning of the corresponding line), and uncomment the line using the circular microphone array on line 11 as the input device (see red box in the referenced image).

4) Press ESC to return to command mode, then type :wq and press Enter to save and exit.

```
COMMAND  wakeup_demo.py
:wq
```

5) In the terminal, run the wake-up program with the following command:

```
python3 ~/large_models/wakeup_demo.py
```

```
> python3 ~/large_models/wakeup_demo.py
```

6) After about 30 seconds of initialization, speak the wake word “hello hiwonder” to test the device.

```
www.hiwonder.com
sh: 1: pinctrl: not found
start...
hello hiwonder
```

3. Brief Program Overview

This is a Python-based wake word detection script that utilizes the speech module to process audio input and detect a specific wake word (e.g., “HELLO_HIWONDER”).

Source Code Path: /home/ubuntu/large_models/wakeup_demo.py

3.1. Importing Required Modules

```
import os
import time
from speech import awake
```

os: Used for handling file paths and executing system-level commands.

time: Provides delay functions to prevent overly frequent detection attempts.

speech: The core module responsible for processing audio input and

detecting the wake word.

3.2. Initializing the wonderecho Class

```
port = '/dev/ttyUSB0'
kws = awake.WonderEchoPro(port)
```

3.3. Attempts to Turn Off the Cooling Fan on Raspberry Pi 5

```
try:
    os.system('pinctrl FAN_PWM a0')
except:
    pass
```

Purpose: Attempts to turn off the cooling fan by executing the system command **pinctrl FAN_PWM op dh**. This helps minimize background noise from the fan that could interfere with wake word detection.

Error Handling: If the command fails (e.g., due to unsupported hardware), the program catches the exception and continues running without interruption.

3.4. Main Wake Word Detection Loop

```
kws.start() # 开始检测
print('start...')
```

The program starts the wake word detection thread using kws.start().

It prints start... to indicate that detection has been successfully initiated.

3.5. Main Program Logic

```
while True:
    try:
        if kws.wakeup(): # 检测到唤醒
            print('hello hiwonder')
        time.sleep(0.02)
    except KeyboardInterrupt:
        kws.exit() # 关闭处理
        try:
            os.system('pinctrl FAN_PWM a0')
        except:
            pass
        break
```

During each iteration, the program checks whether the wake word has been detected. If the wake word is detected, it prints keyword detected.

The detection frequency is controlled using time.sleep(0.02) to prevent excessive CPU usage.

Pressing Ctrl+C triggers a KeyboardInterrupt, which gracefully exits the detection loop.

Upon exit, the program calls kws.exit() to stop the wake word detection process.

The fan is then restored to its original state (if applicable).

4. Extended Functionality

4.1. Modifying the Wake-Up Response Text

In this section, you'll learn how to change the message that appears after a successful wake word detection.

1) For example, if the wake word "HELLO_HIWONDER" is detected, and you'd like the program to print "hello" instead of the default message, follow the steps below. Navigate to the large_models directory and open the script with:

```
vim  wakeup_demo.py
```

2) Press i to enter INSERT mode (you'll see -- INSERT -- at the bottom of the screen). Locate the line '**print('hello hiwonder')**', and modify it to 'print('hello')'

```
i
```

```
print('start...')
while True:
    try:
        if kws.wakeup(): # 检测到唤醒
            print('hello')
        time.sleep(0.02)
```

3) Press ESC, then type :wq and press Enter to save and exit.

```
:wq
```

```
~
:wq
```

4) Finally, run the program with:

```
python3 wakeup_demo.py
```

```
start...
hello
```

4.2. Creating Custom Firmware for WonderEchoPro

If you'd like to create more advanced or customized wake words and voice commands, please refer to the document titled:
“Appendix → 1. Firmware Flashing Tool → Creating Firmware for WonderEchoPro”.