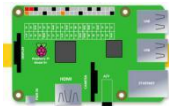
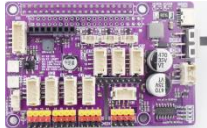



Lesson 20 Local Speech Recognition

20.1 Overview

In this lesson, we will guide beginners to learn how the Raspberry Pi achieves localized speech recognition. We will also conduct an in-depth analysis of its principles, helping everyone to form a basic understanding of offline speech recognition.

20.2 Required Components

Components	Quantity	Picture
Raspberry Pi	1	
Adeept Robot HAT V3.1	1	
Microphone Module	1	

20.3 Principle Introduction

Speech recognition consists of three main parts: acoustic model, language model, and decoder. These three parts work together to enable the computer to accurately recognize and convert speech signals into text or commands. Speech recognition technology is an interdisciplinary field that combines knowledge from multiple fields such as psychology, physiology, acoustics, and linguistics.

The working principle of speech recognition includes steps such as speech signal acquisition, preprocessing, feature extraction, pattern matching, and result output. The following will explain the working principles and related technologies of each step:

- Voice signal acquisition

Sound input device: converts the user's speech waveform into an analog electrical signal through devices such as microphones, and then converts it into a digital signal through an analog-to-digital converter (ADC).

Sampling and quantization: According to the Nyquist sampling law, analog signals are sampled and quantized to ensure that digital signals can accurately represent the original speech.

- Preprocessing

Noise reduction processing: Eliminate background noise and improve the quality of speech signals. This can be achieved through various filters, such as low-pass filters, high pass filters, and band-pass filters.

Endpoint detection: Determine the starting and ending positions of speech signals for subsequent processing. Short term energy and short-term average zero crossing rate are usually used to detect the boundaries of speech signals.

- Feature Extraction

Linear Predictive Cepstral Coefficients: Extracting feature parameters from speech signals using LPC technology, which can describe the fundamental features of speech.

Mel frequency cepstral coefficients: Based on the auditory characteristics of the human ear, feature parameters are extracted through a filter bank model and discrete Fourier transform, and are widely used in modern speech recognition systems.

- Pattern matching

Acoustic model: Hidden Markov Model (HMM) is a mainstream method that calculates the degree of matching between speech signals and models through state transitions and observation probabilities.

Language model: The N-gram model is used to calculate the probability of word sequences appearing, helping to determine the most likely recognition result. In recent years, deep neural networks such as RNNLM have also been widely used for training language models.

- Result output

Decoding and synthesis: Combining the scores of acoustic and language models, using search algorithms to find the best word sequence, and finally outputting the recognition results as text or executing corresponding commands.

Sherpa ncnn speech recognition:

Main functions: speech recognition, streaming speech recognition. Speak while recognizing. No need to access the network, no need for data transmission, fully local recognition.

Recognition effect: The recognition speed is fast and the effect is good, but it only supports WAV format audio. Other formats need to be converted before recognition.

20.4 Demonstration

1. Install the dependency library, and then enter the following command in the command window:

```
sudo apt-get install -y swig portaudio19-dev python3-all-dev python3-pyaudio flac
```

```
pi@raspberrypi:~$ sudo apt-get install -y swig portaudio19-dev python3-all-dev python3-pyaudio flac
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
swig is already the newest version (4.1.0-0.2).
portaudio19-dev is already the newest version (19.6.0-1.2).
python3-all-dev is already the newest version (3.11.2-1).
python3-pyaudio is already the newest version (0.2.13-1).
flac is already the newest version (1.4.2+ds-2).
0 upgraded, 0 newly installed, 0 to remove and 273 not upgraded.
```

```
sudo apt-get install alsa-utils libasound2-dev
```

```
pi@raspberrypi:~ $ sudo apt-get install alsa-utils libasound2-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
alsa-utils is already the newest version (1.2.8-1+rpt1).
Suggested packages:
  libasound2-doc
The following NEW packages will be installed:
  libasound2-dev
0 upgraded, 1 newly installed, 0 to remove and 305 not upgraded.
Need to get 110 kB of archives.
After this operation, 676 kB of additional disk space will be used.
```

```
sudo apt-get install -y git cmake
```

```
pi@raspberrypi:~ $ sudo apt-get install -y git cmake
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.39.5-0+deb12u2).
cmake is already the newest version (3.25.1-1).
0 upgraded, 0 newly installed, 0 to remove and 273 not upgraded.
```

2. Build **sherpa-ncnn** directly on Raspberry Pi.

Firstly, we need to clone the repository named sherpa-ncnn from GitHub to Raspberry Pi.

```
sudo git clone https://github.com/k2-fsa/sherpa-ncnn
```

```
pi@raspberrypi:~ $ sudo git clone https://github.com/k2-fsa/sherpa-ncnn
Cloning into 'sherpa-ncnn'...
remote: Enumerating objects: 2300, done.
remote: Counting objects: 100% (890/890), done.
remote: Compressing objects: 100% (222/222), done.
remote: Total 2300 (delta 761), reused 669 (delta 668), pack-reused 1410 (from 2)
Receiving objects: 100% (2300/2300), 2.21 MiB | 83.00 KiB/s, done.
Resolving deltas: 100% (1287/1287), done.
```

Switch to the newly cloned project directory **sherpa-ncnn**

```
cd sherpa-ncnn
```

```
pi@raspberrypi:~ $ cd sherpa-ncnn
pi@raspberrypi:~/sherpa-ncnn $
```

Create a folder named '**build**' in the project directory to store the compiled files

```
sudo mkdir build
```

```
pi@raspberrypi:~/sherpa-ncnn $ sudo mkdir build
pi@raspberrypi:~/sherpa-ncnn $
```

```
cd build
```

```
pi@raspberrypi:~/sherpa-ncnn $ cd build
pi@raspberrypi:~/sherpa-ncnn/build $
```

Configure the project's build process

```
sudo cmake \
-DCMAKE_BUILD_TYPE=Release \
-DCMAKE_C_FLAGS="-march=armv7-a -mfloat-abi=hard -mfpu=neon" \
-DCMAKE_CXX_FLAGS="-march=armv7-a -mfloat-abi=hard -mfpu=neon" \
..
```

```
pi@raspberrypi:~/sherpa-ncnn/build $ sudo cmake \
-DCMAKE_BUILD_TYPE=Release \
-DCMAKE_C_FLAGS="-march=armv7-a -mfloat-abi=hard -mfpu=neon" \
-DCMAKE_CXX_FLAGS="-march=armv7-a -mfloat-abi=hard -mfpu=neon" \
..
-- The C compiler identification is GNU 12.2.0
-- The CXX compiler identification is GNU 12.2.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
```

Use the **make** tool for compilation. The **-j6** option enables up to 6 compilation tasks to run concurrently, thereby enhancing the compilation speed.

```
sudo make -j6
```



```

pi@raspberrypi:~/sherpa-ncnn/build $ sudo make -j6
[ 1%] Built target ncnn-generate-spirv
[ 1%] Building C object _deps/portaudio-build/CMakeFiles/portaudio_static.dir/src/common/pa_converters.c.o
[ 2%] Building CXX object _deps/kaldi_native_fbank-build/kaldi-native-fbank/csrc/CMakeFiles/kaldi-native-fbank-core.dir/feature-fbank.cc.o
[ 2%] Building C object _deps/portaudio-build/CMakeFiles/portaudio_static.dir/src/common/pa_allocation.c.o
[ 3%] Building C object _deps/portaudio-build/CMakeFiles/portaudio_static.dir/src/common/pa_cpuload.c.o
[ 3%] Building CXX object _deps/kaldi_native_fbank-build/kaldi-native-fbank/csrc/CMakeFiles/kaldi-native-fbank-core.dir/feature-functions.cc.o
[ 4%] Building CXX object _deps/kaldi_native_fbank-build/kaldi-native-fbank/csrc/CMakeFiles/kaldi-native-fbank-core.dir/feature-window.cc.o
[ 4%] Building CXX object _deps/kaldi_native_fbank-build/kaldi-native-fbank/csrc/CMakeFiles/kaldi-native-fbank-core.dir/fftsg.cc.o
[ 4%] Building C object _deps/portaudio-build/CMakeFiles/portaudio_static.dir/src/common/pa_debugprint.c.o
[ 5%] Building C object _deps/portaudio-build/CMakeFiles/portaudio_static.dir/src/common/pa_dither.c.o
[ 5%] Building C object _deps/portaudio-build/CMakeFiles/portaudio_static.dir/src/common/pa_front.c.o
[ 6%] Building CXX object _deps/kaldi_native_fbank-build/kaldi-native-fbank/csrc/CMakeFiles/kaldi-native-fbank-core.dir/mel-computations.cc.o

```

After construction, you will find executable files in the directory: **bin**

```
ls -lh bin/
```

```

pi@raspberrypi:~/sherpa-ncnn/build $ ls -lh bin/
total 16M
-rwxr-xr-x 1 root root 2.6M Mar 13 06:15 decode-file-c-api
-rwxr-xr-x 1 root root 2.5M Mar 13 06:15 generate-int8-scale-table
-rwxr-xr-x 1 root root 2.6M Mar 13 06:15 sherpa-ncnn
-rwxr-xr-x 1 root root 2.6M Mar 13 06:15 sherpa-ncnn-alsa
-rwxr-xr-x 1 root root 2.7M Mar 13 06:15 sherpa-ncnn-microphone
-rwxr-xr-x 1 root root 2.5M Mar 13 06:15 sherpa-ncnn-vad

```

Switch to the **sherpa-ncnn** directory first, and then download the voice model.

```
cd ../
```

```
sudo wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13.tar.bz2
```

```

pi@raspberrypi:~/sherpa-ncnn/build $ cd ../
sudo wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13.tar.bz2
--2025-03-13 06:18:12-- https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13.tar.bz2
Resolving github.com (github.com)... 20.205.243.166
Connecting to github.com (github.com)|20.205.243.166|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/532519562/bc3ad08d-e13a-4422-960b-0a68c4c76a9e?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250313%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250313T061813Z&X-Amz-Expires=300&X-Amz-Signature=1a5ad6eedc172b1b6b9fb2c98dd3d5f7e1fbfe99550cedd6a30a6dd938bf220b&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dsherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13.tar.bz2&response-content-type=application%2Foctet-stream [following]

```

Decompress the downloaded speech model.

```
sudo tar xvf sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13.tar.bz2
```

```

pi@raspberrypi:~/sherpa-ncnn $ sudo tar xvf sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13.tar.
bz2
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-pnnx.ncnn.bin
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/README.md
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/.gitattributes
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.param
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/test_wavs/
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/test_wavs/3.wav
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/test_wavs/0.wav
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/test_wavs/4.wav
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/test_wavs/2.wav
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/test_wavs/5.wav
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/test_wavs/1.wav
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-pnnx.ncnn.param
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/export-for-ncnn-bilingual.sh
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.bin
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.bin
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.param
sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/tokens.txt

```

3.The environment required for using ncnn has now been configured, and you can start the speech recognition. Your voice will be recognized and the result will be output to the console.

Real time speech recognition through microphone

```

./build/bin/sherpa-ncnn-microphone \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/tokens.txt \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-
pnnx.ncnn.param \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-
pnnx.ncnn.bin \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-
pnnx.ncnn.param \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-
pnnx.ncnn.bin \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-
pnnx.ncnn.param \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-
pnnx.ncnn.bin

```

```

pi@raspberrypi:~/sherpa-ncnn $ ./build/bin/sherpa-ncnn-microphone \
./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/tokens.txt \
./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-
pnnx.ncnn.param \
./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-
pnnx.ncnn.bin \
./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-
pnnx.ncnn.param \
./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-
pnnx.ncnn.bin \
./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-
pnnx.ncnn.param \
./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-
pnnx.ncnn.bin
RecognizerConfig(featurizer=FeatureExtractorConfig(sampling_rate=16000, feature
_dim=80), model_config=ModelConfig(encoder_param="./sherpa-ncnn-streaming-zipfor
mer-bilingual-zh-en-2023-02-13/encoder_jit_trace-pnnx.ncnn.param", encoder_bin="
./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-p
nnx.ncnn.bin", decoder_param="./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-
2023-02-13/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./sherpa-ncnn-stream
ing-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.bin", joine
r_param="./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit
_trace-pnnx.ncnn.param", joiner_bin="./sherpa-ncnn-streaming-zipformer-bilingual
-zh-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.bin", tokens="./sherpa-ncnn-streami
ng-zipformer-bilingual-zh-en-2023-02-13/tokens.txt", encoder_num_threads=4, deco

```

If speech recognition cannot be achieved using the microphone command, then **sherpa-ncnn-alsa** can be used.

Real time speech recognition through **sherpa-ncnn-alsa**

For instance, if the output is:

You can use the command **`arecord -l`** to check the sequence number of your sound card device.

```
arecord -l
```

```

pi@raspberrypi:~ $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 2: WebCamera [WebCamera], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0

```

If you want to select **card 2** and the device 0 on that card, please use:

plughw:2,0

In the example, a USB recording device is connected, and the sequence number of the sound card is 2. plughw:2,0


```

sudo ./build/bin/sherpa-ncnn-alsa \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/tokens.txt \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-
pnnx.ncnn.param \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-
pnnx.ncnn.bin \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-
pnnx.ncnn.param \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-
pnnx.ncnn.bin \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-
pnnx.ncnn.param \

./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-
pnnx.ncnn.bin \

plughw:2,0 \

4 \

greedy_search

```

```

pi@raspberrypi:~/sherpa-ncnn $ sudo ./build/bin/sherpa-ncnn-alsa ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/tokens.txt ./sherpa-ncnn-st
reaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-pnnx.ncnn.param ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_tra
ce-pnnx.ncnn.bin ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.param ./sherpa-ncnn-streaming-zipformer-bilingual
-zh-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.bin ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.param ./sherpa-nc
nn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.bin plughw:2,0 4 greedy_search
RecognizerConfig(featurizer=FeatureExtractorConfig(sampling_rate=16000, feature_dim=80), model_config=ModelConfig(encoder_param="./sherpa-ncnn-streaming-zip
former-bilingual-zh-en-2023-02-13/encoder_jit_trace-pnnx.ncnn.param", encoder_bin="./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_t
race-pnnx.ncnn.bin", decoder_param="./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./sherpa-nc
nn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-1
3/joiner_jit_trace-pnnx.ncnn.param", joiner_bin="./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.bin", tokens="./sher
pa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/tokens.txt", encoder_num_threads=4, decoder_num_threads=4, joiner_num_threads=4), decoder_config=Decod
erConfig(method="greedy_search", num_active_paths=4), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=2
.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_cont
ain_nonsilence=False, min_trailing_silence=0, min_utterance_length=300))), enable_endpoint=True, hotwords_file="greedy_search", hotwords_score=1.5)
Current sample rate: 16000
Recording started!
Use recording device: plughw:2,0
0: 测试一下语音识别
1: out
2: kiss
3: forward
4: backward

```

The **sherpa-ncnn-alsa** program is used to achieve low-latency streaming speech recognition based on the ALSA audio driver, which is suitable for Linux systems. The main differences from the previous **sherpa-ncnn-microphone** are as follows:

Audio Interface: It directly controls the audio device using ALSA (alsa-lib) instead of PortAudio (a cross-platform library).

Device Selection: The input source is specified through the ALSA device name (such as **plughw:2,0**), which is suitable for scenarios of embedded Linux or when precise control of the sound card is required.

greedy_search: The decoding method can choose **greedy_search** or **modified_beam_search**. Greedy search is faster in speed, but may not be as accurate as beam search; Beam search is more accurate, but slower. You can choose the appropriate decoding method according to your needs.

20.5 Speech Recognition Implementation Code

The use of speech recognition involves three files, namely **Speech_Recognition.py**, **Speech.py**, and **Text.py**. Among them, **Speech_Recognition.py** is responsible for initiating the execution of the other two files. The **Speech.py** file is responsible for starting the speech recognition function and outputting the results to the **output.txt** file. The **Text.py** file is responsible for reading speech recognition results from the **output.txt** file. text.py file and print it to the terminal.

Speech_Recognition.py

```
01  #!/usr/bin/env/python
02  # File name   : Speech_Recognition.py
03  # Website    : www.Adeept.com
04  # Author     : Adeept
05  # Date      : 2025/03/13
06  import subprocess
07  import time
08  # Define the paths to two Python programs to run
09  program1_path = "./Speech.py"
10  program2_path = "./Text.py"
11
12  # Create two sub processes and run two Python programs separately
13  process1 = subprocess.Popen(["python3", program1_path])
14  time.sleep(3)           # Waiting for speech recognition to start
```

```

15 process2 = subprocess.Popen(["python3", program2_path])
16
17 # Waiting for two child processes to complete
18 process1.wait()
19 process2.wait()

```

Speech.py

```

01 #!/usr/bin/env/python
02 # File name   : Speech.py
03 # Website    : www.Aadept.com
04 # Author     : Aadept
05 # Date      : 2025/03/13
06 import os
07
08 def main():
09     # cmd = "sudo /home/pi/sherpa-ncnn/build/bin/sherpa-ncnn-microphone \
10     #       /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/tokens.txt
11     \
12     #       /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-
13     13/encoder_jit_trace-pnnx.ncnn.param \
14     #       /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-
15     13/encoder_jit_trace-pnnx.ncnn.bin \
16     #       /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-
17     13/decoder_jit_trace-pnnx.ncnn.param \
18     #       /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-
19     13/decoder_jit_trace-pnnx.ncnn.bin \
20     #       /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-
21     13/joiner_jit_trace-pnnx.ncnn.param \
22     #       /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-
23     13/joiner_jit_trace-pnnx.ncnn.bin"
24
25     # You can also use the `sherpa-ncnn-alsa` command for speech recognition. Note that you need to
26     replace the `plughw:3,0` parameter with the serial number of your own sound card.
27     cmd = "sudo /home/pi/sherpa-ncnn/build/bin/sherpa-ncnn-alsa \
28     /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/tokens.txt
29     \
30     /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-
31     13/encoder_jit_trace-pnnx.ncnn.param \
32     /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-
33     13/encoder_jit_trace-pnnx.ncnn.bin \
34     /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-
35     13/decoder_jit_trace-pnnx.ncnn.param \
36     /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-
37     13/decoder_jit_trace-pnnx.ncnn.bin \
38     /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-
39     13/joiner_jit_trace-pnnx.ncnn.param \
40     /home/pi/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-
41     13/joiner_jit_trace-pnnx.ncnn.bin \
42     plughw:3,0 \
43     4 \
44     greedy_search"
45
46     os.system(f"{cmd} > output.txt 2>&1") #Run a command-line program and save the output results to a
47     file named 'output. txt'

```

```
if __name__ == "__main__":  
    main()
```

Text.py

```
01 #!/usr/bin/env/python  
02 # File name   : Text.py  
03 # Website    : www.Aadept.com  
04 # Author     : Aadept  
05 # Date      : 2025/03/13  
06 import time  
07  
08 file_position = 0  
09 while True:  
10     with open("output.txt", "r") as file: # Read the file named "output.txt"  
11         file.seek(file_position)  
12         new_lines = file.readlines() # Read all lines from the current file pointer position to the end  
13     of the file  
14         if new_lines:  
15             for line in new_lines:  
16                 if "Started" in line:  
17                     print(line.split("Started")[-1].strip() + "\n")  
18                     elif file_position > 0: # Ensure we print lines after the first "Started"  
19                         print(line.strip())  
20                     file_position = file.tell()  
21             time.sleep(3) # Read every 3 second
```

You can perform speech recognition by running the following command.

```
sudo cd aadept_picar-b2/examples/11_Speech_Recognition/  
sudo python3 Speech_Recognition.py
```