

ros noeticでのpyopenjtalkを用いたテキスト音声変換

目標は、デイサービスで働くロボット。Ryzen9のggmlで作成した回答をpyopenjtalkを用いてテキスト音声変換し、ros topicにsudioデータとして書き込み、T8で音声を再生する。

構成

・T8\$	Client PC CF-T8	Debian10
・noetic2\$	T8 Docker Container	
・Ryzen9\$	Server PC Minis Forum HX90	Debian12
・noetic\$	Ryzen9 Docker Container	

目次

pyopenjtalk実行環境構築

必要なパッケージをインストール

pyopenjtalkをインストール

[noetic yo@Ryzen9:~\\$ pip3 install --no-cache-dir pyopenjtalk](#)

[noetic yo@Ryzen9:~\\$ pip3 install soundfile](#)

動作確認

scriptファイル作成・実行

[作成: noetic yo@Ryzen9:~\\$ vi pyopenjtalk_test.py](#)

[実行: noetic yo@Ryzen9:~\\$ python3 pyopenjtalk_test.py](#)

ros noeticでのpyopenjtalkの活用

パッケージの作成しnodeを追加・実行

[パッケージ: noetic yo@Ryzen9:~/catkin_ws/src\\$ catkin create_pkg ros_pyopenjtalk rospy roscpp std_msgs](#)

[ノード: noetic](#)

[yo@Ryzen9:~/catkin_ws/src/ros_pyopenjtalk/scripts\\$ vi chatgpt_pyopenjtalk01.py](#)

T8音声出力

T8で音声出力用のnode作成

[noetic2 yo@T8:~/catkin_ws/src/audio_t8/scripts\\$ audio_player_node.py](#)

Ryzen9テキスト音声変換後、T8で音声出力

[Ryzen9 docker noeticでpyopenjtalk起動](#)

[noetic yo@Ryzen9:~\\$ rosrn ros pyopenjtalk
chatgpt_pyopenjtalk01.py](#)

[T8 docker noetic_audio_rev2で音声出力audio_player_node.py起動](#)

[noetic2 yo@T8:~\\$ rosrn audio_t8 audio_player_node.py](#)

[感想](#)

[今後の予定](#)

[参考 生成AI](#)

[rostopicを購読し、pyopenjtalkにより音声に変換](#)

[質問](#)

[回答: ChatGPT](#)

[回答: Gemini](#)

[音声出力](#)

[質問](#)

[回答: ChatGPT](#)

pyopenjtalk実行環境構築

 20241221pyopenjtalkによるテキスト音声変換 を参考にros melodicにpyopenjtalkの実行環境を構築する。

必要なパッケージをインストール

```
apt-get update && apt-get install -y --no-install-recommends \  
  build-essential \  
  libsndfile1-dev \  
  libpulse-dev \  
&& apt-get clean \  
&& rm -rf /var/lib/apt/lists/*
```

pyopenjtalkをインストール

```
noetic yo@Ryzen9:~$ pip3 install --no-cache-dir pyopenjtalk
```

```
noetic yo@Ryzen9:~$ pip3 install --no-cache-dir pyopenjtalk  
Collecting pyopenjtalk
```

```
  Downloading pyopenjtalk-0.3.4.tar.gz (1.4 MB)
```

| 1.4 MB 1.5

MB/s

Installing build dependencies ... done

Getting requirements to build wheel ... done

Installing backend dependencies ... done

Preparing wheel metadata ... done

Collecting importlib-resources; python_version < "3.9"

Downloading importlib_resources-6.4.5-py3-none-any.whl (36 kB)

Collecting numpy>=1.20.0

Downloading

numpy-1.24.4-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)

| 17.3 MB 1.5

MB/s

Requirement already satisfied: tqdm in

./local/lib/python3.8/site-packages (from pyopenjtalk) (4.67.1)

Collecting zipp>=3.1.0; python_version < "3.10"

Downloading zipp-3.20.2-py3-none-any.whl (9.2 kB)

Building wheels for collected packages: pyopenjtalk

Building wheel for pyopenjtalk (PEP 517) ... done

Created wheel for pyopenjtalk:

filename=pyopenjtalk-0.3.4-cp38-cp38-linux_x86_64.whl

size=5966710 sha256=767cff4dfa10795e02a144074e660a8867b79
bcc9a65b80a3f6f6be04389105be

Stored in directory:

/tmp/pip-ephem-wheel-cache-pkj5hjom/wheels/7c/c9/dd/0a5288337
dabf1cfe8e5c9e50ca30812a7f63c0451b1b85c80

Successfully built pyopenjtalk

Installing collected packages: zipp, importlib-resources, numpy,
pyopenjtalk

WARNING: The scripts f2py, f2py3 and f2py3.8 are installed in
'/home/yo/.local/bin' which is not on PATH.

Consider adding this directory to PATH or, if you prefer to suppress
this warning, use --no-warn-script-location.

Successfully installed importlib-resources-6.4.5 numpy-1.24.4
pyopenjtalk-0.3.4 zipp-3.20.2

noetic yo@Ryzen9:~\$ pip3 install soundfile

Collecting soundfile

Downloading soundfile-0.13.0-py2.py3-none-any.whl (25 kB)

Requirement already satisfied: cffi>=1.0 in

./local/lib/python3.8/site-packages (from soundfile) (1.17.1)

Requirement already satisfied: numpy in

./local/lib/python3.8/site-packages (from soundfile) (1.24.4)

Requirement already satisfied: pycparser in

./local/lib/python3.8/site-packages (from cffi>=1.0->soundfile) (2.22)

Installing collected packages: soundfile

Successfully installed soundfile-0.13.0

動作確認

scriptファイル作成・実行

作成: *noetic yo@Ryzen9:~\$ vi pyopenjtalk_test.py*

```
import pyopenjtalk
```

```
import numpy as np
```

```
import soundfile as sf
```

```
# 音声合成を実行
```

```
text = "こんにちは、pyopenjtalkのテストです。"
```

```
waveform, sr = pyopenjtalk.tts(text)
```

```
# 音量を下げる
```

```
volume_factor = 0.00001 # 0.0~1.0の範囲で設定
```

```
wave = waveform * volume_factor
```

```
# 音声をWAVファイルとして保存
```

```
sf.write("output.wav", wave, sr)
```

```
print("音声ファイルが生成されました: output.wav")
```

実行: *noetic yo@Ryzen9:~\$ python3 pyopenjtalk_test.py*

Downloading:

"https://github.com/r9y9/open_jtalk/releases/download/v1.11.1/open_jtalk_dic_utf_8-1.11.tar.gz"


```

def text_to_speech_callback(msg):
    """Callback function to handle text data and publish audio."""
    text = msg.data
    rospy.loginfo(f"Received text: {text}")

    # Convert text to speech using pyopenjtalk
    waveform, _ = pyopenjtalk.tts(text)

    # 音量を下げる
    volume_factor = 0.00001 # 0.0~1.0の範囲で設定
    wav = waveform * volume_factor

    # Convert numpy array to raw audio bytes
    audio_data = (wav * 32767).astype(np.int16).tobytes()

    # Publish the audio data
    audio_msg = AudioData()
    audio_msg.data = audio_data
    audio_pub.publish(audio_msg)

    rospy.loginfo("Published audio data.")

def main():
    """Main function to initialize the ROS node and handle
    subscriptions/publishers."""
    rospy.init_node('text_to_speech_node')

    # Subscriber to the text topic
    rospy.Subscriber(INPUT_TOPIC, String,
    text_to_speech_callback)

    # Publisher for the audio topic
    global audio_pub
    audio_pub = rospy.Publisher(OUTPUT_TOPIC, AudioData,
    queue_size=10)

    rospy.loginfo("Text-to-Speech node started.")

```



```

        rate=self.rate,
        output=True)

    # Subscribe to audio topic
    rospy.Subscriber('audio_pyopenjtalk', AudioData,
self.audio_callback)
    rospy.loginfo("Audio player node initialized.")

def audio_callback(self, msg):
    """Callback to handle incoming audio data."""
    rospy.loginfo("Received audio data.")
    audio_data = np.frombuffer(msg.data, dtype=np.int16)
    self.stream.write(audio_data.tobytes())

def cleanup(self):
    """Cleanup resources on shutdown."""
    self.stream.stop_stream()
    self.stream.close()
    self.audio.terminate()
    rospy.loginfo("Audio player node shut down.")

def run(self):
    """Run the ROS event loop."""
    rospy.on_shutdown(self.cleanup)
    rospy.spin()

if __name__ == '__main__':
    try:
        player = AudioPlayerNode()
        player.run()
    except rospy.ROSInterruptException:
        pass

```

Ryzen9テキスト音声変換後、T8で音声出力

全てのプログラムを起動する必要があるので、手間である。いずれ roslaunch に纏める。

1. Ryzen9 docker noetic で ros vosk 起動

2. T8 docker noetic_audio_rev2でaudio_captre起動

3. Ryzen9 docker noeticでggmlを起動

noetic yo@Ryzen9:~/catkin_ws/src/ros_vosk/launch\$ roslaunch

ros_vosk audio_ros_vosk0chatgpt.launch

noetic2 yo@T8:~\$ roslaunch audio_t8 audio_capture.launch

noetic yo@Ryzen9:~\$ rosruncatgpt02_node.py

ここまでは、 20250106ros_noeticにggml環境構築と同じ。

4. Ryzen9 docker noeticでpyopenjtalk起動

5. T8 docker noetic_audio_rev2で音声出力audio_player_node.py起動

Ryzen9 docker noeticでpyopenjtalk起動

```
noetic yo@Ryzen9:~$ rosruncatgpt_pyopenjtalk01.py
```

```
[INFO] [1736917175.203292]: Text-to-Speech node started.
```

```
[INFO] [1736917207.953706]: Received text:
```

テスターです。ちなみに、このテスターの定価は1,500円でした。テスターの針が0の位置にくるように調整して、電源をONにします。そして、電源をONにして1

分待ちます。1分たったら電源をOFFにしてテスターをコンセントから抜きます。そして、テスターの電源を入れてテスターが反応した場所の電圧を測ります。結

果は、10Vでした。

```
[INFO] [1736917211.068382]: Published audio data.
```

T8 docker noetic_audio_rev2で音声出力audio_player_node.py起動

```
noetic2 yo@T8:~$ rosruncatgpt_audio_player_node.py
```

```
ALSA lib pcm_dsnoop.c:641:(snd_pcm_dsnoop_open) unable to open slave
```

```
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.rear
```

```
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
```

ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM
cards.pcm.side
ALSA lib pcm_route.c:869:(find_matching_chmap) Found no
matching channel map
ALSA lib pcm_route.c:869:(find_matching_chmap) Found no
matching channel map
ALSA lib pcm_route.c:869:(find_matching_chmap) Found no
matching channel map
ALSA lib pcm_oss.c:377:(_snd_pcm_oss_open) Unknown field port
ALSA lib pcm_oss.c:377:(_snd_pcm_oss_open) Unknown field port
ALSA lib pcm_usb_stream.c:486:(_snd_pcm_usb_stream_open)
Invalid type for card
ALSA lib pcm_usb_stream.c:486:(_snd_pcm_usb_stream_open)
Invalid type for card
[INFO] [1736917017.200133]: Audio player node initialized.
[INFO] [1736917066.446708]: Received audio data.
[INFO] [1736917110.222640]: Received audio data.
ALSA lib pcm.c:8526:(snd_pcm_recover) underrun occurred
ALSA lib pcm.c:8526:(snd_pcm_recover) underrun occurred
[INFO] [1736917147.576472]: Received audio data.
ALSA lib pcm.c:8526:(snd_pcm_recover) underrun occurred
[INFO] [1736917188.120164]: Received audio data.
ALSA lib pcm.c:8526:(snd_pcm_recover) underrun occurred
[INFO] [1736917245.267838]: Received audio data.
ALSA lib pcm.c:8526:(snd_pcm_recover) underrun occurred
[INFO] [1736917371.232929]: Received audio data.
ALSA lib pcm.c:8526:(snd_pcm_recover) underrun occurred
[INFO] [1736917400.889867]: Received audio data.
ALSA lib pcm.c:8526:(snd_pcm_recover) underrun occurred
[INFO] [1736917541.072983]: Received audio data.
ALSA lib pcm.c:8526:(snd_pcm_recover) underrun occurred

感想

T8でマイク入力からRyzen9のggmlを利用し回答を作成し、rostopicにテキストで出力、pyopnejtalkによるテキスト音声変換、T8での再生まで繋がった。

通常の私の開発スピードからすると、異常なスピードである。まっ、コードは生成AIが作ってくれるから当たり前か。ただし、小さなエラーもあり、それを修復する必要がある。そこを強調することも無いのだが。

今後の予定

6. T8の液晶にロボットの顔を表示し、ggml変換中に考え中の顔を表示する
7. servoを使った機体のデザイン決定
9. ros noeticで構成されるプログラムの起動をroslaunchで纏める
10. 全体の流れを再考する(このままだと、スピーカの音をマイクが拾ってしまう)ヘッドフォン？

参考 生成AI

rostopicを購読し、pyopenjtalkにより音声に変換

質問

ros noeticにおいて、topic(/gpt_neox/result)テキストデータを購読し、pyopenjtalkを用いて音声に変換し、audio_common topicに書き込みを行いたい。

また、購読するテキストデータのトピック名は、(/gpt_neox/result)とします。

出力する音声データのトピック名を、audio_pyopenjtalkとしてください。

回答: ChatGPT

以下は、ROS Noeticで `/gpt\_neox/result` トピックからテキストデータを購読し、`pyopenjtalk` を使用して音声に変換し、`audio\_common` パッケージを使用して音声データを `audio\_pyopenjtalk` トピックにパブリッシュするノードのPythonコード例です。

まず、必要なパッケージをインストールしてください:

```
```bash
pip install pyopenjtalk
sudo apt install ros-noetic-audio-common-msgs
```
```

そして以下のコードを作成します。

```
import rospy
from std_msgs.msg import String
from audio_common_msgs.msg import AudioData
import pyopenjtalk
import numpy as np
import wave
import io

def text_to_speech_callback(msg):
    """Callback function to handle text data and publish audio."""
    text = msg.data
    rospy.loginfo(f"Received text: {text}")

    # Convert text to speech using pyopenjtalk
    wav, _ = pyopenjtalk.tts(text)

    # Convert numpy array to raw audio bytes
    audio_data = (wav * 32767).astype(np.int16).tobytes()

    # Publish the audio data
    audio_msg = AudioData()
    audio_msg.data = audio_data
    audio_pub.publish(audio_msg)

    rospy.loginfo("Published audio data.")

def main():
    """Main function to initialize the ROS node and handle
    subscriptions/publishers."""
    rospy.init_node('text_to_speech_node')

    # Subscriber to the text topic
    rospy.Subscriber('/gpt_neox/result', String,
text_to_speech_callback)

    # Publisher for the audio topic
```

```
global audio_pub
audio_pub = rospy.Publisher('audio_pyopenjtalk', AudioData,
queue_size=10)
```

```
rospy.loginfo("Text-to-Speech node started.")
```

```
rospy.spin()
```

```
if __name__ == '__main__':
    try:
        main()
    except rospy.ROSInterruptException:
        pass
```

説明

1. **トピック購読**:

`/gpt\_neox/result` トピックから `String` 型のメッセージを購読します。

2. **音声変換**:

`pyopenjtalk.tts()` を使用して、テキストを音声データ (numpy 配列) に変換します。

3. **音声データの加工**:

生成された音声データを 16-bit PCM に変換し、`AudioData` メッセージ形式に変換します。

4. **トピックへの公開**:

変換した音声データを `audio\_pyopenjtalk` トピックにパブリッシュします。

ノードの起動

上記コードを保存し、ROSのワークスペース内に配置した後、以下の手順で起動します。

1. **ビルド**:

必要に応じてROSのパッケージをビルドします。

2. **起動**:

ノードを以下のコマンドで起動します。

```
``bash
roslaunch your_package_name text_to_speech_node.py
``
```

注意点

- `pyopenjtalk` はテキストの内容に基づいて日本語音声を生じます。英語テキストを入力する場合は、意図した結果が得られない可能性があります。

- `audio\_common\_msgs` を利用するために、依存関係として `audio\_common\_msgs` を `package.xml` と `CMakeLists.txt` に追加してください。

回答: Gemini

ROS Noetic環境で/gpt_neox/resultトピックのテキストデータを購読し、pyopenjtalkで音声に変換、audio_pyopenjtalkトピックに書き込む方法について説明します。以下の手順とコード例を参考にしてください。

1. 必要なパッケージのインストール

まず、必要なパッケージをインストールします。

Bash

```
sudo apt update
sudo apt install ros-noetic-rospy ros-noetic-std-msgs
ros-noetic-audio-common
pip install pyopenjtalk
```

2. Pythonスクリプトの作成 (text_to_speech.py)

以下のPythonスクリプトを作成します。

Python

```
#!/usr/bin/env python
```

```
import rospy
from std_msgs.msg import String
from audio_common_msgs.msg import AudioData
import pyopenjtalk
import numpy as np
import wave
import io
```

```
class TextToSpeechNode:
```

```
    def __init__(self):
```

```
        rospy.init_node('text_to_speech', anonymous=True)
```

```
        self.subscriber = rospy.Subscriber('/gpt_neox/result', String,
self.callback)
```

```
        self.publisher = rospy.Publisher('audio_pyopenjtalk', AudioData,
queue_size=10)
```

```
        self.sampling_rate = 16000 # pyopenjtalkのデフォルトサンプリン
グレート
```

```
    def callback(self, data):
```

```
        text = data.data
```

```
        try:
```

```
            # pyopenjtalkで音声合成
```

```
            wav_data, _ = pyopenjtalk.tts(text)
```

```
            # numpy arrayからbytesに変換
```

```
            wav_bytes = (wav_data * 32767).astype(np.int16).tobytes()
```

```
            # AudioDataメッセージを作成してpublish
```

```
            audio_msg = AudioData()
```

```
            audio_msg.data = wav_bytes
```

```
            self.publisher.publish(audio_msg)
```

```
            rospy.loginfo("Published audio data.")
```

```
    except Exception as e:
```

```
        rospy.logerr("Error during speech synthesis: %s", e)
```

```
if __name__ == '__main__':  
    try:  
        node = TextToSpeechNode()  
        rospy.spin()  
    except rospy.ROSInterruptException:  
        pass
```

3. スクリプトの実行権限付与

作成したスクリプトに実行権限を与えます。

Bash

```
chmod +x text_to_speech.py
```

4. ROSパッケージの作成 (必要に応じて)

もしROSパッケージ内で実行する場合は、以下の手順でパッケージを作成し、スクリプトを配置します。

Bash

```
cd ~/catkin_ws/src  
catkin_create_pkg text_to_speech_pkg rospy std_msgs  
audio_common_msgs  
cd text_to_speech_pkg  
mkdir scripts  
mv ~/path/to/text_to_speech.py scripts/  
CMakeLists.txtに以下を追加します。
```

CMake

```
add_executable(text_to_speech scripts/text_to_speech.py)  
target_link_libraries(text_to_speech ${catkin_LIBRARIES})  
package.xmlに以下を追加します。
```

XML

```
<build_depend>message_generation</build_depend>
```

```
<exec_depend>message_runtime</exec_depend>  
<depend>audio_common_msgs</depend>
```

catkin_makeを実行します。

5. 実行

ROS Coreを起動し、作成したノードを実行します。

Bash

roscore

roslaunch text_to_speech_pkg text_to_speech

/gpt_neox/resultトピックにテキストメッセージをpublishすると、
audio_pyopenjtalkトピックに音声データがpublishされます。

重要なポイント:

サンプリングレート: pyopenjtalkのデフォルトサンプリングレートは16000Hzです。AudioDataメッセージを使用する際には、このレートに合わせる必要があります。

エラー処理: スクリプトには簡単なエラー処理が含まれていますが、必要に応じてより詳細なエラー処理を追加してください。

音声の再生: audio_commonパッケージには音声再生ツールが含まれています。roslaunch sound_play soundplay_node.pyを実行後、rostopic pub /audio_pyopenjtalk audio_common_msgs/AudioData "data: \$(base64 /path/to/your/audio.wav)"などで再生できますが、今回の場合はrostopic echo /audio_pyopenjtalkでデータを確認し、必要に応じてファイルに保存して再生すると良いでしょう。

文字コード: テキストの文字コードによっては正しく音声合成できない場合があります。必要に応じてUTF-8に変換するなど文字コードの処理を行ってください。

この手順で、ROS Noetic環境でテキストから音声への変換を行うことができます。不明な点があれば、遠慮なく質問してください。

音声出力

質問

前項で作成したnodeから書き込まれた音声データaudio_pyopenjtalkトピックを購読して、音声出力を行うnodeをpythonで作成してください。

回答: ChatGPT

```
import rospy
from audio_common_msgs.msg import AudioData
import pyaudio
import numpy as np

class AudioPlayerNode:
    def __init__(self):
        """Initialize the audio player node."""
        rospy.init_node('audio_player_node')

        # Audio settings
        self.rate = 48000 # Default rate for pyopenjtalk
        self.channels = 1
        self.format = pyaudio.paInt16

        # PyAudio instance
        self.audio = pyaudio.PyAudio()

        # Audio stream
        self.stream = self.audio.open(format=self.format,
                                       channels=self.channels,
                                       rate=self.rate,
                                       output=True)

        # Subscribe to audio topic
        rospy.Subscriber('audio_pyopenjtalk', AudioData,
self.audio_callback)
        rospy.loginfo("Audio player node initialized.")

    def audio_callback(self, msg):
        """Callback to handle incoming audio data."""
```

```

    rospy.loginfo("Received audio data.")
    audio_data = np.frombuffer(msg.data, dtype=np.int16)
    self.stream.write(audio_data.tobytes())

def cleanup(self):
    """Cleanup resources on shutdown."""
    self.stream.stop_stream()
    self.stream.close()
    self.audio.terminate()
    rospy.loginfo("Audio player node shut down.")

def run(self):
    """Run the ROS event loop."""
    rospy.on_shutdown(self.cleanup)
    rospy.spin()

if __name__ == '__main__':
    try:
        player = AudioPlayerNode()
        player.run()
    except rospy.ROSInterruptException:
        pass

```

説明

- **購読設定**:**
`audio\_pyopentalk` トピックを購読し、音声データ (`AudioData` 型) を受信します。
- **音声データの再生**:**
`pyaudio` を使用して、受信した音声データを再生します。データは 16-bit PCMフォーマットのため、`numpy` を使用して適切にデコードします。
- **終了処理**:**
ノードの終了時に `PyAudio` のリソースをクリーンアップします。

使用方法

1. 上記コードをファイルに保存します (例: `audio\_player\_node.py`)。

2. 必要なパッケージをインストールします:

```
```bash
pip install pyaudio numpy
```
```

3. ノードを起動します:

```
```bash
roslaunch your_package_name audio_player_node.py
```
```

4. 他のノードから `audio\_pyopenjtalk` トピックに音声データがパブリッシュされると、このノードで音声が出力されます。

注意点

- ****サンプリングレート****: このコードでは `pyopenjtalk` のデフォルトサンプリングレートである 48kHz を使用しています。異なるサンプリングレートの場合は適宜変更してください。

- ****リアルタイム性****: 音声データが遅延なく再生されるよう、トピックのキューサイズを調整してください。