fonctions vocales avec le Raspberry Pi Zero

Donner de la voix à votre projet

Somnath Bera (Inde)

Explorons la reconnaissance vocale (STT) et la synthèse vocale (TTS) sur le Raspberry Pi Zero, avec différentes options logicielles, des instructions d'installation et des applications potentielles pour des projets intégrant le traitement audio et la commande vocale.

La synthèse vocale (TTS) et la conversion de la parole en texte (STT) sont les deux principales fonctions vocales que nous allons explorer avec le Raspberry Pi Zero, un ordinateur monocarte équipé d'un processeur performant cadencé à 1 GHz, qui peut être surcadencé à 1,3 GHz avec un dissipateur thermique en métal pour un refroidissement passif (pour un surcadençage jusqu'à 1,7 GHz, un refroidissement actif est nécessaire). En termes de performances, cette carte offre des performances comparables au Raspberry Pi 4, bien qu'elle dispose d'une mémoire limitée à 512 MB. J'ai testé plusieurs logiciels de TTS et STT sur cette carte, et je présente les résultats détaillés ci-dessous, ainsi que des idées de projets pour inspirer les lecteurs.

Surcadençage

Comme le Pi Zero dispose d'une RAM limitée (seulement 512 MB), le surcadençage fournit une augmentation significative des performances, ce qui est particulièrement utile pour les tâches de TTS et STT. Ce gain en puissance et en efficacité vient toutefois au prix d'une augmentation potentielle de la chaleur et d'une possible instabilité du CPU. Sur le Raspberry Pi Zero, l'overclocking est simple. En ajoutant seulement deux lignes au fichier */boot/config.txt*, en sauvegardant et en redémarrant, vous pouvez immédiatement bénéficier d'une hausse des performances. Il est essentiel de surveiller la température du CPU ; avec un bon dissipateur thermique en métal, il est possible de surcadencer en toute sécurité

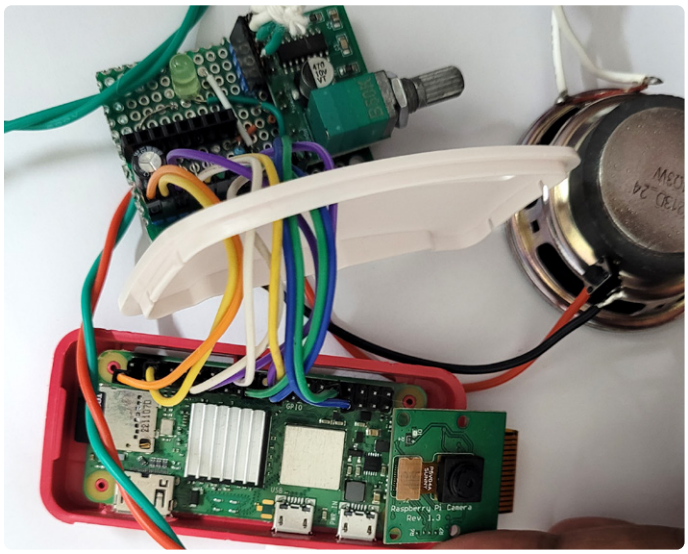

Figure 1. Gros plan sur le Raspberry Pi Zero avec son dissipateur thermique métallique.

jusqu'à 1,3 GHz, en maintenant une température de 65°C à 70°C sans compromettre la stabilité. Pour des fréquences supérieures, comme jusqu'à 1,7 GHz, un refroidissement actif devient nécessaire. Référez-vous à la **figure 1** qui montre le dissipateur métallique que j'ai utilisé

Éditez le fichier *config.txt*, par exemple avec l'éditeur Nano :

```
sudo nano /boot/config.txt
```

et ajoutez ces deux lignes :

```
arm_freq=1300
over_voltage=2
```

Si vous souhaitez essayer l'overclocking jusqu'à 1,7 GHz, augmentez le dernier paramètre à 4 ou 5. Puis enregistrez et redémarrez pour appliquer les modifications. Pour surveiller la température du processeur, vous pouvez utiliser la commande `vcgencmd measure_temp`.

Sortie son sur le Raspberry Pi Zero

Le Raspberry Pi Zero ne dispose pas de connecteur intégré pour la sortie audio. Pour obtenir du son, vous devez connecter un périphérique HDMI, tel qu'un téléviseur. Cependant, il existe des adaptateurs HDMI vers AV qui peuvent séparer le signal audio. Vous pouvez également utiliser deux des broches GPIO compatibles PWM (12, 13, 18, 19) pour activer la sortie audio en ajoutant une seule ligne au fichier /boot/config.txt. Ouvrez le fichier avec `sudo nano /boot/config.txt`, puis ajoutez la ligne suivante :

```
dtoverlay=audremap,pins_18_13
```

Sauvegardez le fichier et redémarrez ; la sortie stéréo sur les broches 18 et 13 est maintenant activée.
Ces broches émettent des signaux carrés modulés en largeur d'impulsion de niveau logique (3,3 V), qui ne conviennent pas pour alimenter directement des haut-parleurs, mais qui peuvent être suffisants pour une petite paire d'écouteurs. Vous pouvez utiliser un module d'amplification audio, tel que le PAM8403. J'ai réalisé le montage présenté dans la **figure 2**. Les signaux droit et gauche sont filtrés par des filtres RC passe-bas, puis couplés en courant alternatif aux entrées de l'ampli via les deux condensateurs de 10 µF. Vous êtes maintenant prêt à profiter d'une sortie stéréo de qualité ! Vous pouvez voir le circuit en fonctionnement sur [1], où j'utilise le Raspberry Pi Zero pour diffuser une chanson bouddhiste. Le montage complet est illustré dans la **figure 3**.

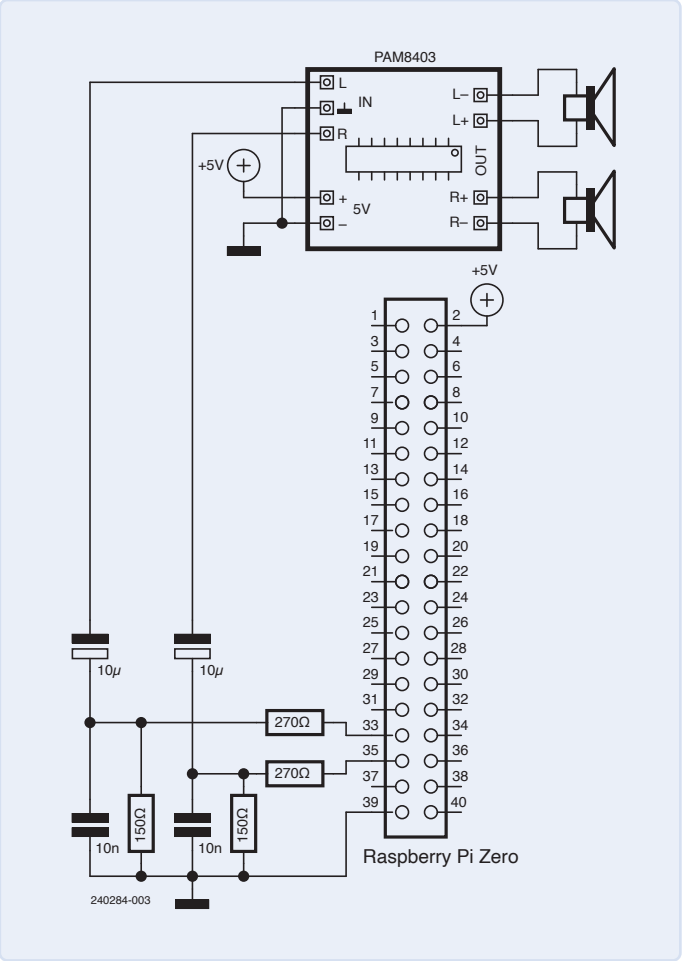

Figure 2. Schéma du projet

Logiciel de synthèse vocale

Examinons quelques-unes des options offertes par les logiciels de synthèse vocale.

- **eSpeak / eSpeak-ng :** eSpeak et sa nouvelle version eSpeak-ng (nouvelle génération) sont des logiciels de synthèse vocale open-source matures, disponibles pour les systèmes basés sur Linux (comme Debian) et Mac OS. Ces programmes sont faciles à installer et peuvent être utilisés immédiatement. Pour installer eSpeak-ng sur le Pi Zero, il suffit de suivre les instructions fournies, en utilisant les commandes :

```
sudo apt-get install espeak-ng
espeak-ng -f temp.txt
espeak-ng "Hello Readers, welcome to this project"
```

Vous devriez alors entendre la phrase « Hello Readers... ». Outre l'anglais, eSpeak-ng supporte plusieurs langues et voix. `espeak-ng -v en+f2 "Input text"` émettra une voix féminine. Pour plus de détails, consultez le manuel avec `man espeak-ng`.

- **Google Text-To-Speech (gTTS) :** gTTS offre des voix beaucoup plus naturelles que eSpeak ou eSpeak-ng. Je l'utilise souvent dans mes projets d'IA. Vous pouvez l'installer :

```
pip install gtts playsound
```

Voir le **listage 1** pour un exemple de code Python pour démarrer.

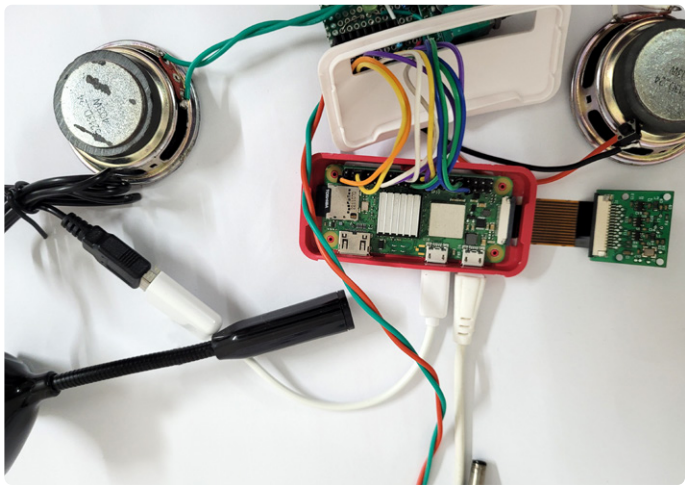

Figure 3. Configuration complète avec microphone, amplificateur et haut-parleurs.

Entrée vocale sur le Pi Zero

La sortie vocale fonctionne parfaitement jusqu'à présent. Pour l'entrée vocale, un microphone USB économique (environ 2 $ chez *robu.in*) est parfaitement fonctionnel et prêt à l'emploi. Après l'avoir

Listage 1. Utilisation de la synthèse vocale de Google.

```
from gtts import gTTS
import subprocess

def text_to_speech(text):
    # Use gTTS to convert text to speech and save to an audio file
    tts = gTTS(text=text, lang="en")
    tts.save("output.mp3")
    # Use mpg123 to play the audio file
    subprocess.run(["mpg123", "-q", "output.mp3"])

if __name__ == "__main__":
    while True:
        # Get the string to speak
        question = "Please ask your question:"
        text_to_speech(question)
```

branché, la commande `lsusb` vous permettra de vérifier que le microphone est détecté. Voici ce que j'obtiens avec cette commande dans mon exemple :

```
Bus 001 Device 002: ID 4c4a:4155 Jieli Technology UACDemoV1.0
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Vous aurez peut-être besoin d'un câble OTG pour faire l'interface entre le microUSB (côté Pi Zero) et l'USB (microphone). Maintenant que le microphone est connecté, pour le tester complètement, essayez cette commande dans un terminal :

```
arecord -f S16_LE -r 33100 -d 10 -c 1 output.wav
```

Les paramètres sont expliqués ci-dessous.

- `-f S16_LE` : spécifie le format audio. `S16_LE` signifie signé 16 bits little-endian.
- `-r 33100` : Définit le taux d'échantillonnage à 33 100 Hz.
- `-d 10` : spécifie la durée de l'enregistrement en secondes (par exemple, 10 s dans cet exemple).
- `-c 1` : Définit le nombre de canaux audio à 1 (mono). Pour les enregistrements stéréo, utilisez `-c 2`.
- `output.wav` : spécifie le nom du fichier de sortie (dans ce cas, *output.wav*).

Après avoir exécuté cette commande, `arecord` capturera l'audio à un taux d'échantillonnage de 33 100 Hz pendant 10 s et l'enregistrera en tant que fichier output.wav dans le répertoire actuel. Vous pouvez modifier des paramètres tels que la durée (`-d`), le nom du fichier de sortie et d'autres paramètres en fonction de vos besoins spécifiques.

Pour lire l'audio enregistré, exécutez `aplay output.wav`. Veillez à régler la position du microphone et à l'éloigner du haut-parleur pour garantir des enregistrements audio nets.
Jusqu'ici, tout fonctionne bien ! Vous avez configuré votre source audio, installé le logiciel de synthèse vocale sur le Pi Zero et essayé avec succès l'enregistrement audio et la conversion du texte en parole avec eSpeak, eSpeak-ng ou gTTS. Passons maintenant à la conversion de la parole en texte.

Logiciels de transcription de la parole

Google offre un service payant pour la transcription de la parole nécessitant une connexion internet. En outre, pour la synthèse vocale, Google propose également une API en ligne qui est gratuite jusqu'à une certaine limite, après quoi un compte payant est nécessaire.
Dans le cadre de ce projet, sur le Pi Zero, il est préférable d'utiliser un logiciel de synthèse vocale hors ligne, à condition que la parole soit suffisamment claire. Bien qu'il existe une fonction de Google (voir ci-dessous), mais il y a aussi deux excellents logiciels libres disponibles pour la conversion parole-texte : spchcat et Pocketsphinx. Alors que spchcat est basé sur une ligne de commande, Pocketsphinx est une solution basée sur une bibliothèque. Les deux logiciels peuvent être utilisés avec Python en utilisant différentes méthodes. Le module subprocess est utilisé pour appeler spchcat, tandis que le module pocketsphinx peut être installé directement dans Python avec `pip`. Voici la procédure d'installation de ces deux outils performants.

Spchcat

Spchcat est un logiciel open-source qui utilise le modèle de reconnaissance TensorFlow. Il est actuellement compatible avec Linux et Raspberry Pi et prend en charge 46 modèles de langues, y compris des langues indiennes telle que le tamoul et le bengali, ma langue

maternelle. Vous pouvez télécharger le logiciel et les instructions depuis [2].
Spchcat est un outil en ligne de commande qui traite l'audio à partir de fichiers WAV. C'est un programme volumineux, environ 1,2 Go, qui ne peut pas être directement instillé sur un Pi Zero. Pour contourner cette limitation, suivez les étapes suivantes :

1. Téléchargement : commencez par télécharger le logiciel *spchcat* depuis son dépôt GitHub sur votre ordinateur.
2. Transfert : utilisez scp (*secure copy*) pour transférer le fichier téléchargé sur la carte SD de votre Pi Zero.
3. Installation : comme le Pi Zero ne peut pas gérer l'installation, retirez la carte SD et insérez-la dans un Raspberry Pi B+ ou un Raspberry Pi 4.
4. Installation sur le Pi 4 : double-cliquez sur le logiciel téléchargé sur le Raspberry Pi 4. Le processus d'installation prendra environ 25 minutes.
5. Dernière étape : après l'installation, replacez la carte SD sur votre Pi Zero. Le logiciel *spchcat* est maintenant prêt à être utilisé sur votre Pi Zero.

Grâce à cette méthode, vous pouvez surmonter les contraintes de capacité du Pi Zero.
Après l'installation, vous pouvez commencer à capturer de l'audio via le microphone par défaut en lançant le logiciel. Les résultats s'afficheront directement dans le terminal :

```
spchcat
```

Si vous ne disposez pas de microphone ou si vous souhaitez transcrire de l'audio provenant d'un autre programme, vous pouvez utiliser l'argument `--source` avec la valeur `system`. Cette option tentera d'écouter l'audio joué par votre machine, y compris les vidéos ou les chansons, et de transcrire toute parole détectée :

```
spchcat --source=system
```

Lancez une vidéo YouTube ou jouez un fichier audio sur votre système. Vous verrez la transcription vocale apparaître dans la fenêtre du terminal au fur et à mesure de la lecture.
Pour traiter un fichier WAV avec spchcat, utilisez la commande suivante :

```
spchcat /home/bera/myaudio.wav
```

Après le traitement, le texte transcrit s'affichera dans le terminal. Pour choisir la langue de transcription, utilisez l'argument `--language`, par exemple pour l'allemand ou l'anglais, utilisez :

```
spchcat --language=de_DE
spchcat --language=en_US
```

Attention, spchcat est gourmand en ressources et peut surchauffer le processeur. La surveillance de la température du processeur est

Liste des composants

La plupart de ces composants, à l'exception du Raspberry Pi Zero, sont disponibles à des prix très modérés sur les sites Internet chinois courants.

- Raspberry Pi Zero 2 W
- Module d'amplification PAM8403
- Haut-parleurs de 4 ohms (2 unités)
- Microphone USB
- Câble USB OTG

cruciale. Il est essentiel de surveiller la température du CPU et d'utiliser un dissipateur thermique de haute qualité sur le Pi Zero pour gérer efficacement la chaleur produite lors de son fonctionnement.
Vous pouvez tester spchcat avec des fichiers WAV d'exemple sans avoir à les créer vous-même. Téléchargez des fichiers WAV depuis [3] et utilisez-les pour évaluer les résultats de la transcription avec votre Pi Zero. Ils vous aideront à juger la qualité d'enregistrement requise pour une transcription précise.
Pour enregistrer la transcription produite par la commande dans un fichier, la méthode typique sous Linux est d'utiliser le caractère `>` :

```
spchcat your-audio-file.wav > /tmp/transcript.txt
```

Enfin, pour utiliser spchcat dans un script Python, vous devez utiliser le module subprocess. Assurez-vous que votre environnement utilise Python 3.7 ou plus. Si subprocess n'est pas préinstallé, vous pouvez l'installer avec `pip install subprocess`.
Sur le Pi Zero, spchcat est capable de traiter jusqu'à 4 minutes d'enregistrement et de transcription de fichiers WAV. Sur un Raspberry Pi 4, il peut effectuer des transcriptiosn en continu à partir de sources telles que des vidéos YouTube. La qualité de l'enregistrement est cruciale pour obtenir des résultats précis. Placer le microphone près de la source sonore améliore généralement la précision de la transcription.

PocketSphinx

PocketSphinx est un moteur de reconnaissance vocale léger et open-source développé par l'université Carnegie Mellon. Il traite l'audio mono canal 16 bits PCM provenant de l'entrée standard ou de fichiers et tente de reconnaître la parole en utilisant des modèles acoustiques et linguistiques par défaut. Plus simple et moins complexe que *Spchcat*, *PocketSphinx*, il supporte plusieurs langues, dont l'anglais, le chinois, le français, l'espagnol, l'allemand et le russe. Cependant, je ne l'ai testé qu'avec l'anglais. *PocketSphinx* peut convertir des fichiers WAV en texte directement, mais ce processus peut s'avérer lent lorsqu'il est exécuté avec Python.
Pour installer *PocketSphinx* ainsi que le modèle de langue anglaise, utilisez les commandes suivantes :

```
pip install pocketsphinx
sudo apt-get install -y python3-pocketsphinx
pocketsphinx-en-us
```

En Python, la classe `LiveSpeech` du module *PocketSphinx* permet la reconnaissance vocale. La valeur du time-out, ici fixée à 1,0, peut être ajustée selon les besoins. Voici un programme Python très simple pour tester *PocketSphinx* :

```
from pocketsphinx import LiveSpeech
speech = LiveSpeech()
speech = LiveSpeech(silence_timeout=1.0)
print("Listening...")
for phrase in speech:
      print(f"Transcript: {phrase}")
```

La transcription en temps réel est possible à la fois sur le Pi Zero et le Raspberry Pi 4. Sur le Pi Zero, il arrive que *PocketSphinx* omette certains mots, alors qu'il fonctionne parfaitement sur le Raspberry Pi 4. J'ai trouvé que la qualité de la transcription offerte par *PocketSphinx* est souvent inférieure à celle de spchcat.

Google Speech-to-Text

Google Speech-to-Text (STT) supporte 125 langues et est très efficace, ce qui en fait une excellente option pour les appareils à capacité limitée, comme le Raspberry Pi Zero. Pour utiliser Google STT, installez la bibliothèque Python *SpeechRecognition*, qui permet de l'utiliser :

```
pip install SpeechRecognition
```

Un programme d'exemple simple d'utilisation de cette bibliothèque, avec Google Speech-To-Text, est présenté dans le **listage 2.**
La bibliothèque SpeechRecognition a l'avantage de supporter plusieurs moteurs de reconnaissance vocal en plus de celui de Google. Par exemple, Pour utiliser *PocketSphinx*, vous pouvez réutiliser le code du listage 2 en remplaçant `text = recognizer.recognize_google(audio)` par `text = recognizer.recognize_sphinx(audio)` à la ligne 15.
Le **listage 3** montre comment utiliser cette bibliothèque avec le moteur de reconnaissance vocale de Google pour transcrire du texte à partir d'un fichier. Je vous recommande vivement de le tester ! Bien entendu, il est possible de combiner l'enregistrement audio et la transcription dans un même script Python. Par exemple, le **listage 4** illustre comment enregistrer de l'audio pendant dix secondes tout en posant une question, puis affiche la transcription textuelle de cette question.
J'ai utilisé avec succès Speech-to-Text (STT) et Text-to-Speech (gTTS)

Listage 2. Utilisation de la bibliothèque SpeechRecognition avec Python.

```
import speech_recognition as sr
def speech_to_text():
     # Initialize the recognizer
     recognizer = sr.Recognizer()
     # Use the default microphone as the audio source
     with sr.Microphone() as source:
         print("Speak something...")
         audio = recognizer.listen(source)
     try:
         print("Recognizing...")
         # Use Google Web Speech API to perform speech recognition
         text = recognizer.recognize_google(audio)
         return text
     except sr.UnknownValueError:
         print("Could not understand audio")
         return ""
     except sr.RequestError as e:
         #print(f"Error: ")
         return ""
if __name__ == "__main__":
     while True:
         # Call the speech_to_text function to get text from speech
         result = speech_to_text()
         print(f"You said: ")
```

Listage 3. Reconnaissance vocale depuis un fichier audio.

```
import argparse
import speech_recognition as sr
def speech_to_text(audio_file):
     # Initialize the recognizer
     recognizer = sr.Recognizer()
     # Load the audio file
     with sr.AudioFile(audio_file) as source:
         audio_data = recognizer.record(source)
     try:
         # Use Google Speech Recognition for speech-to-text
         text = recognizer.recognize_google(audio_data)
         return text
     except sr.UnknownValueError:
         print("Google Speech Recognition could not understand the audio")
         return ""
    except sr.RequestError as e:
         print(f"Could not request results from Google Speech Recognition service; ")
         return ""
if __name__ == "__main__":
     # Create argument parser
     parser = argparse.ArgumentParser(description="Convert audio file to text using Google Speech
Recognition")
     parser.add_argument("audio_file", help="Path to the audio file")
     parser.add_argument("destination_file", help="Path to save the text output")
     # Parse command-line arguments
     args = parser.parse_args()
     # Call the speech_to_text function to convert speech to text
     result = speech_to_text(args.audio_file)
     if result:
         # Save the recognized text to the destination file
         with open(args.destination_file, "w") as f:
             f.write(result)
         print(f"Recognized text saved to ")
     else:
         print("Speech recognition failed.")
```

de Google dans divers projets avec le Raspberry Pi Zero, y compris dans des projets basés sur des microprocesseurs comme le *terminal ESP32 ChatGPT* [4][5], où il génère des réponses vocales via un haut-parleur I2S.

Futurs cas d'utilisation ?

La conversion du texte en parole et inversement est une tâche importante. Les grandes entreprises de médias s'appuient sur des logiciels coûteux et des ordinateurs performants pour effectuer ces tâches en temps réel pendant les diffusions en direct. Bien que l'utilisation d'un Raspberry Pi Zero puisse sembler simpliste, cette carte offre des avantages notables, notamment sa taille compacte, sa capacité à être alimentée avec diverses alimentations 5 V et sa consommation d'énergie réduite.

Voici quelques applications potentielles pour la transcription vocale sur le Pi Zero que je recommande :

- **Encyclopédie parlante :** un terminal mains libres où vous posez une question, la confirmez, et le Pi Zero récupère les réponses d'un service tel que ChatGPT, puis les énonce à haute voix. Tout cela peut être réalisé avec un Pi Zero. En savoir plus : [6].
- **Sortie vocale pour les projets d'IA :** dans les projets d'IA, comme ceux que j'ai développés avec Edge Impulse pour la

Listage 4. Reconnaissance vocale et enregistrement audio.

```
import subprocess
import speech_recognition as sr

def record_audio(file_name, duration=10):
    # Record audio from USB microphone using arecord
    command = ["arecord", "-D", "plughw:1,0", "-f", "S16_LE", "-r", "16000", "-d", str(duration), file_name]
    subprocess.run(command)

def speech_to_text(file_name):
    recognizer = sr.Recognizer()
    # Load the audio file for recognition
    with sr.AudioFile(file_name) as source:
        audio_data = recognizer.record(source)
    text = recognizer.recognize_google(audio_data)
    return text.lower()  # Convert to lowercase for easier comparison

if __name__ == "__main__":
    while True:
        # Record audio for question
        record_audio("question.wav", duration=10)  # Increased duration for longer responses
        speech_text = speech_to_text("question.wav")
        print(speech_text)
```

classification d'objets, les résultats de la classification (par exemple, un nombre ou un nom) peuvent être verbalisés par l'appareil grâce à eSpeak ou eSpeak-ng, vous libérant ainsi de la nécessité d'utiliser un écran. Voir un exemple sous [7].

- **Terminal pour personnes aveugles :** un système permettant à une personne malvoyante de poser des questions, le Raspberry Pi recherchant les réponses sur divers sites, y compris OpenAI, pour fournir des réponses vocales.
- **Robotique : i**déal pour les jouets robotisés à commande vocale, en particulier pour les marchés internationaux, en convertissant les commandes en langue étrangère en anglais pour un traitement ultérieur.
- **Lecteurs de codes-barres portables à commande vocale** : utiles dans les supermarchés, où l'appareil lit les codes-barres et annonce les résultats par via des haut-parleurs. Les capacités multilingues d'eSpeak-ng pourraient être un grand particulièrement bénéfiques.
- **Assistant vocal pour les malentendants :** un assistant vocal portable qui fournit une transcription en temps réel des phrases prononcées pour les personnes malentendantes.
- **Assistant de voyage :** facilite la communication dans les pays étrangers en traduisant et en vocalisant les interactions avec les habitants, comme les chauffeurs de taxi ou les serveurs.
- **Lecteur de musique à interaction vocale :** un lecteur de musique portable qui réagit aux commandes vocales, similaire à Alexa d'Amazon.

Ce ne sont là que quelques exemples parmi tant d'autres possibles. Quelles autres applications impliquant la parole pourriez-vous envisager avec des nano-ordinateurs comme le Raspberry Pi ?

240284-04

À propos de l'auteur

Somnath Bera, ingénieur en mécanique diplômé du Jalpaiguri Govt. Engineering College en Inde, occupe le poste de directeur général chez NTPC, le plus grand producteur d'électricité du pays. Passionné par l'électronique, il a réalisé plus de 60 projets novateurs sur Elektor Labs, dont plus de 10 ont été publiés dans le magazine Elektor. Ses projets sont principalement axés sur des solutions aux enjeux de gestion des déchets et des ressources naturelles. Somnath aime exploiter des approches et des plateformes innovantes telles qu'Arduino, Raspberry Pi et ESP32, intégrant divers capteurs et systèmes sans fil, afin de concevoir des solutions efficaces et économiques.

Questions ou commentaires ?

Envoyez un courriel à l'auteur (berasomnath@gmail.com), ou contactez Elektor (redaction@elektor.fr).

Produits

- **Raspberry Pi Zero 2 WH (avec connecteurs)**
  www.elektor.fr/20952
- **John Allwork, *Programming Voice-controlled IoT Applications with Alexa and Raspberry Pi* (Elektor, 2023)**
  www.elektor.fr/20400

LIENS

[1] Exemple de fichier audio joué sur Raspberry Pi Zero : https://youtu.be/YXC7VlVlX9c
[2] Aide et codes de fichier pour spchcat : https://github.com/petewarden/spchcat
[3] Fichiers WAV d'exemple à tester avec spchcat : https://www.voiptroubleshooter.com/open_speech/
[4] « Terminal ESP32-ChatGPT » sur Elektor Labs : https://www.elektormagazine.fr/labs/esp32-chatgtp-terminal
[5] Somnath Bera, « ESP32 ChatGPT Terminal », édition spéciale IA 2024 d'Elektor : http://www.elektormagazine.fr/230536-04
[6] "Raspberry Pi Zero Talking Encyclopedia" on Elektor Labs: https://www.elektormagazine.com/labs/raspberry-pi-zero-encyclopedia
[7] Exemple : Transcription de texte dans les projets d'IA : https://www.youtube.com/watch?v=C5QrCl4XIJU