GINARD, LONDECHAL
LAMBERSENS, AIMÉ

Spécifications

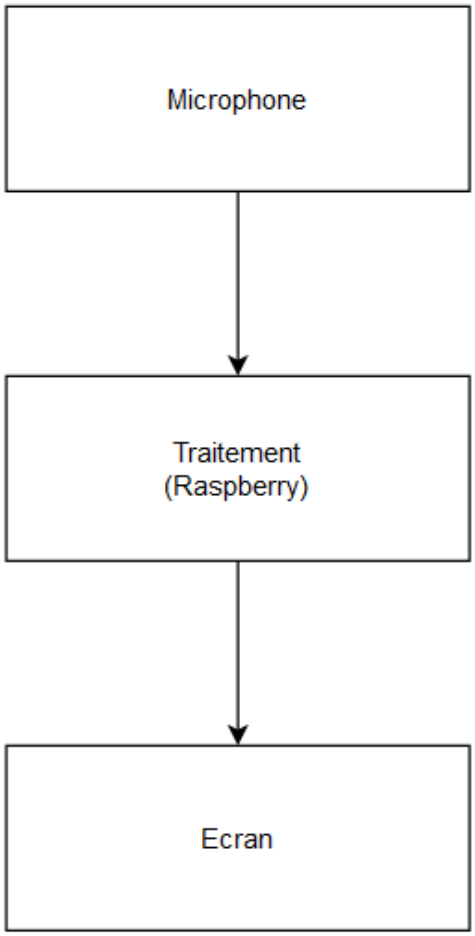
Table des matières

I. Architecture :

Ci-dessous 2 schémas de notre projet. Le premier est le schéma du prototype présenté. Le deuxième est le schéma hypothétique du produit final.

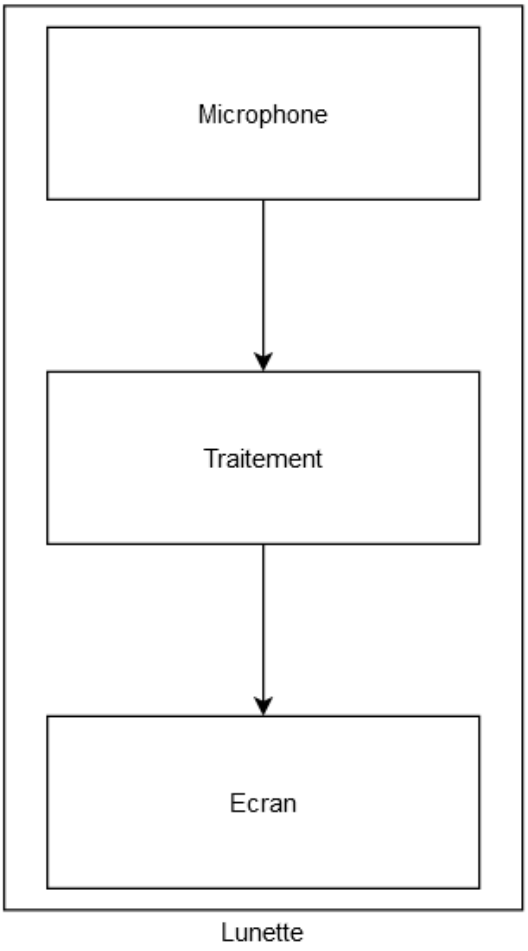
Prototype:

Ci-dessous un schéma simplifié de notre prototype qui utilise une Raspberry Pi 3B :



Produit final :

Ci-dessous un schéma de notre produit final.



La différence notable entre le prototype et le produit final est le support. Pour notre prototype nous avons opté pour un microphone et un petit écran LCD connecté à une Raspberry Pi qui comprend toute la puissance de calcul. Pour notre produit final, nous avons opté pour une monture de lunettes qui comprend une micro-puce pour la puissance de calcul, un petit microphone et un affichage sur les verres des lunettes.

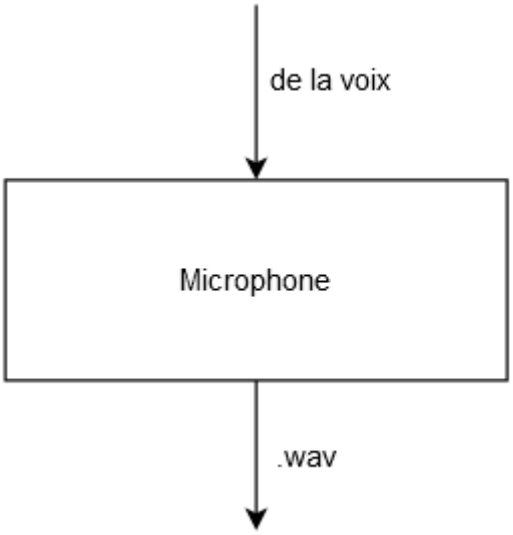
II. Technologie utilisée :

Le langage de programmation utilisé est Python. Nous avons choisi ce langage par soucis de simplicité. Python est un langage que nous maitrisons au sein du groupe ce qui évite d'apprendre de nouveau langage, et donc de gagner du temps. De plus python a l'avantage d'être un langage supporté par les raspbery Pi et par les modules utilisés.

Pour l'ensemble des parties du projet, la librairie "grove.py" est utilisé. Elle offre un support et des fonctionnalités adapté pour les modules de senseur que nous utilisons avec la Raspberry Pi ainsi qu'une documentation officielle (exemple : https://wiki.seeedstudio.com/Grove-LCD_RGB_Backlight/#play-with-raspberry-pi).

2.1. Partie microphone :

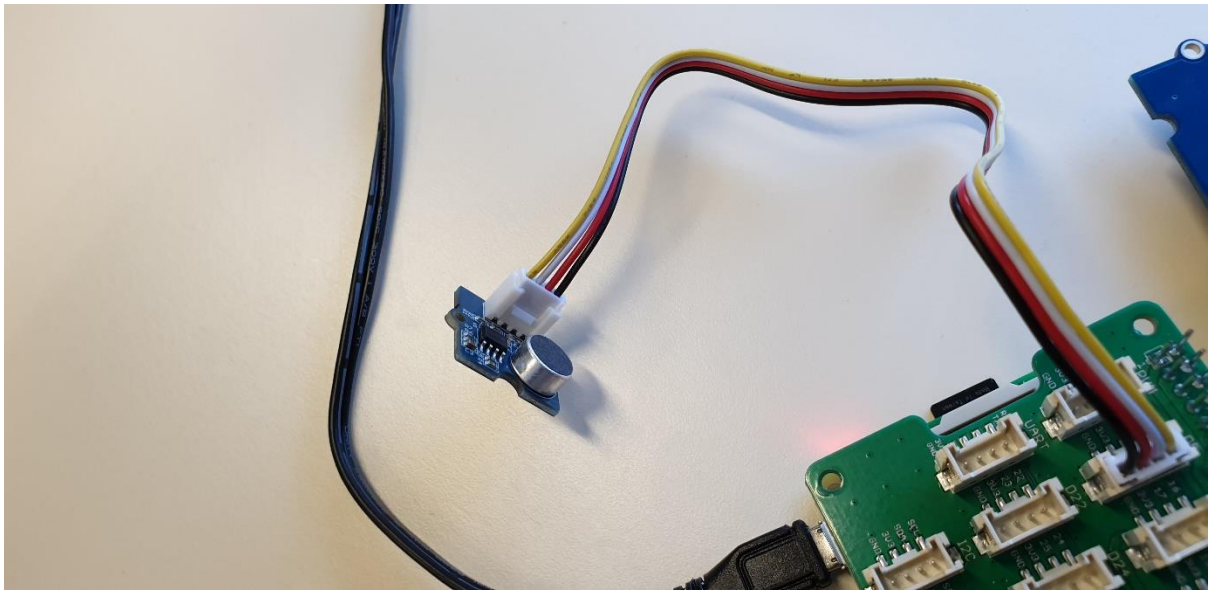
Ci-dessous un schéma pour la partie microphone.



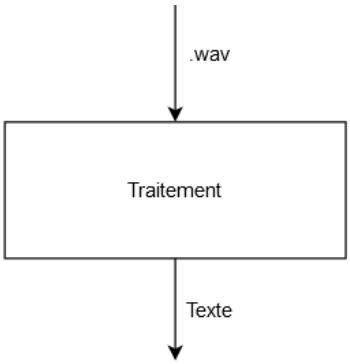
Comme explicité sur le schéma, la voix est enregistrée par le microphone. Cette voix est ensuite envoyée à la partie “traitement” sous forme de court fichier en “.wav”. Le choix de fichier wav à deux intérêts. Premièrement, les fichiers wav sont des conteneurs universels pour les fichiers audio, et la librairie utilisée dans la partie traitement supporte ce format, ce qui simplifie le développement de la partie traitement. Deuxièmement dans notre cas on utilise le codec par défaut qui est en haute-fidélité et donc non destructif ce qui aide la reconnaissance vocale.

Pour la méthode d’enregistrement, nous avons optés pour la sauvegarde de court fichiers wav. En effet, nous avons prévu à terme d’ajouter un module de détection de son qui permet de déterminer la puissance du son. Ce qui veux dire que à chaque augmentation du volume sonore en dB, l’enregistrement se lancerai. Au contraire, à chaque fois que le volume sonore du son diminuerai jusqu’à disparaitre dans le bruit ambiant, l’enregistrement s’arrêterai.

Ci-dessous le module puissance de son utilisé avec la Raspberry Pi :



2.2. Partie traitement :



Comme expliqué précédemment, des fichiers au format wav sont envoyés de la partie microphone vers la partie traitement. La partie traitement analyse ces fichiers .wav pour ensuite les analyser et les retranscrire en format texte. Le texte est ensuite envoyé à la partie affichage.

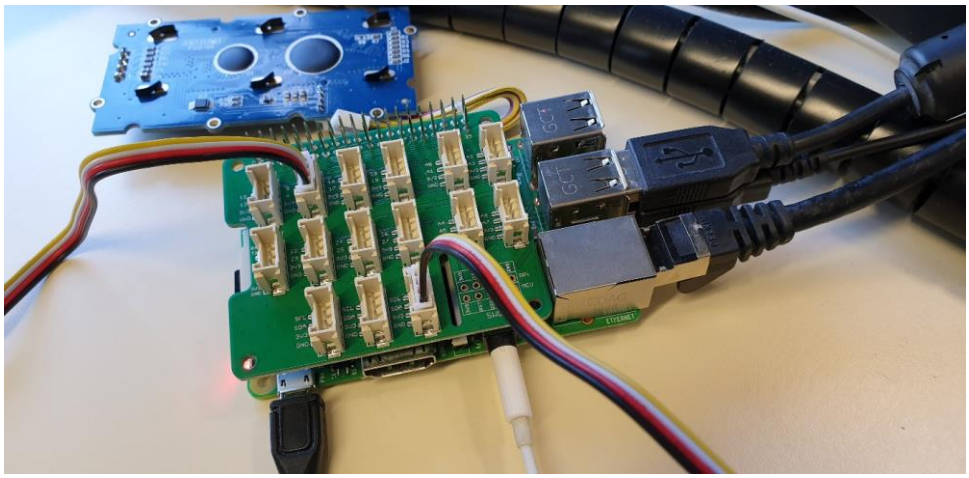
Pour l'analyse des fichiers .wav, une librairie python conséquente est utilisé : "SpeechRecognition". Cette librairie permet, entre autres l'utilisation d'API (en ligne ou hors ligne) utile à la transcription de fichier audio vers du texte. Les API utilisent des modèles d'IA pré-entrainés pour reconnaitre la voix et la passer en texte. L'API que nous avons utilisée est celle mise à disposition par Alphabet (Google). Nous avons choisi cette API pour plusieurs raisons :

- Une API locale (Pocketsphinx) a été testé, mais malheureusement les résultats n'étaient pas concluants avec une grande lenteur de traitement, dû à la faible puissance de calcul de la Raspberry Pi, et avec des résultats très approximatifs.
- L'avantage majeur est que l'API de Google prend en compte toutes les langues, ce qui simplifie le traitement. De plus, des modèles déjà pré-entrainés sont mis à disposition.

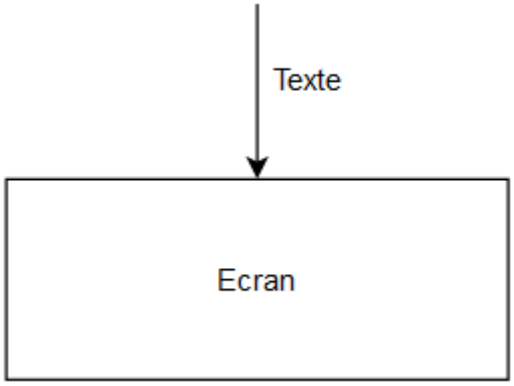
Dans le cadre du produit final, nous pouvons facilement imaginer la création et l'entrainement de notre propre modèle personnalisé. L'utilisation d'un tel modèle à plusieurs avantages :

- Modèle en local, donc non-dépendance au service Google et à une connexion internet constante
- Personnalisation et modification du modèle au fur et à mesure de l'évolution des besoins. Cela permettra de répondre au besoin spécifique de notre produit

Ci-dessous l'image de la Raspberry Pi qui fait le traitement :



2.3. Partie affichage :



Pour finir, voici la dernière partie. Cette partie est très simple. Elle reçoit un flux de caractère en entré et l'affiche sur un écran. Cela implique donc d'échapper les différents caractères spéciaux puis de formater les chaines de caractères au format de l'écran (dans le cas de notre prototype 12 caractères*2 lignes). Pour cela, nous avons fait en sorte de ne pas couper les mots, pour faciliter leurs compréhensions. Donc, à chaque fois qu'un mot est trop long pour rentrer sur ligne de douze caractère nous nous arrêtons à l'espace précèdent. Pour les mots de plus de douze caractères —plus long qu'une ligne donc— le mot est coupé, un tiret de césure est placé et la suite du mot continue à la ligne suivante.

Ci-dessous l'image de notre module d'affichage :

III. Sécurite :

Côté sécurité, chaque lunette possède un ID unique qui lui permettra de se connecter à nos serveurs et d'utiliser les services proposés. La connexion établie entre le serveur et les lunettes est sécurisée par une communication chiffrée. Cela permet de sécuriser les données utilisateur qui transit entre les deux. Il faut tout de même préciser que les lunettes peuvent fonctionner sans connexion internet, ce qui permet, en cas d'attaque, des déconnecter toutes les lunettes pour éviter qu'elles soient impactées.