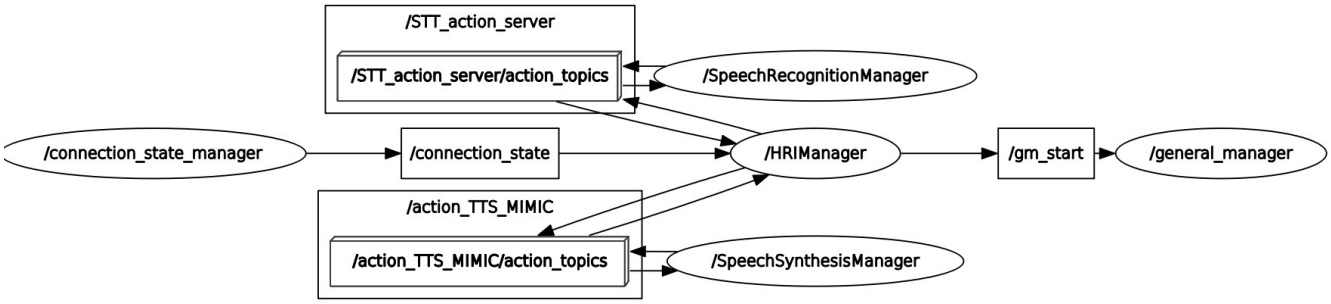
Documentation de la brique technologique d'interaction

1 – HRIManager :

Rôle : Coordonner les actions d'interaction vocale et les affichages des vues sur la tablette

Communications :

- ROS Action avec le GeneralManager pour récupérer les informations du scénario et l'indice de l'étape à afficher sur la tablette (Serveur : HRIManager, Client : GeneralManager via API LTHriManagerPalbator)
- ROS Topic avec le GeneralManager pour transmettre le choix du scénario (Publisher : HRIManager, Subscriber : GeneralManager)
- ROS Action avec le SpeechRecognitionManager pour commander la détection vocale (Serveur : SpeechRecognitionManager, Client : HRIManager)
- ROS Action avec le SpeechSynthesisManager pour commander la synthèse vocale (Serveur : SpeechSynthesisManager, Client : HRIManager)
- ROS Topic avec le connectionManager pour récupérer l'état de connexion Internet (connecté/déconnecté) en temps réel (Publisher : connectionManager, Subscriber : HRIManager)
- SocketIO avec l'application React pour afficher les vues des étapes du scénario sur la tablette



Fonctionnalités implémentées :

- Chargement des vues créées pour les différents scénarios
- Contrôle de la commande de reconnaissance vocale
- Contrôle de la commande de synthèse vocale
- Chargement du menu principal sur la tablette pour choisir le scénario à exécuter via des boutons
- Transmission des données issues de saisie vocale ou manuelle (nom, boisson, âge, pièce à inspecter) au GeneralManager

Fonctionnalités non implémentées :

- Gestion du bouton stop sur la tablette pour arrêter l'exécution du scénario

Configuration :

- Ajout d'un scénario au menu principal → fichier de configuration « config_HRI.yaml » → ajout d'une structure JSON à la liste « list_scenario_available » : un « id » qui sera affiché sur le bouton sur la tablette et un « name » qui correspond au nom du fichier Python du scénario sans l'extension

2 – SpeechRecognitionManager :

Rôle : Exécuter la commande de reconnaissance vocale via PocketSphinx si le robot est déconnecté à Internet ou SpeechRecognition si le robot est connecté.

Communication :

- ROS Action avec le HRIManager pour utiliser la reconnaissance vocale (Serveur : SpeechRecognitionManager, Client HRIManager)

Fonctionnalités implémentées :

- Détection vocale avec PocketSphinx avec micro en live ou lecture de fichier audio
- Détection vocale avec SpeechRecognition avec micro en live ou lecture de fichier audio

Configuration :

- Changement du type d'entrée audio : « Live » ou « AudioFile »
- Ajout d'un nouveau scénario :
    - création d'un dossier au nom du scénario dans le dossier misc du package speechToTextPalbator
    - création d'un dictionnaire de mots détectables :
        - création d'un fichier txt avec un mot par ligne
        - générer le dictionnaire grâce au site http://www.speech.cs.cmu.edu/tools/lmtool-new.html
        - renommer ce dictionnaire et le placer dans le dossier au nom du scénario
        - écrire le chemin relatif du dictionnaire dans le fichier de configuration du package
    - création d'un fichier de mots clés : création d'un fichier texte .kwlist avec sur chaque ligne un mot du dictionnaire séparé d'une tabulation avec le seuil de détection (par exemple : /1e-9/). Pour choisir le seuil de détection correctement, se référer à https://cmusphinx.github.io/wiki/tutoriallm/#using-keyword-lists-with-pocketsphinx. Ecrire le chemin relatif du dictionnaire de mots clés dans le fichier de configuration du package
    - création d'un fichier de grammaire .gram :
        - se référer aux fichiers .gram existants pour la construction du fichier et au lien https://www.w3.org/TR/2000/NOTE-jsgf-20000605/
        - donner un nom à la grammaire (grammar grammar_name)
        - donner un nom à la règle de grammaire (public <rule_name>)
        - reporter les noms du fichier (champ « gram »), de la grammaire (champ « grammar ») et de la règle de grammaire (champ « rule ») dans le fichier de configuration du package

- Création d'un fichier de vérification :
  - Création d'un fichier txt contenant les mots valides compte tenu du contexte de détection (par exemple, uniquement une liste de noms dans le cas d'une détection de nom d'une personne)
  - Reporter le chemin relatif du fichier de vérification dans le fichier de configuration
- Après ces différentes étapes, le fichier de configuration doit ressembler aux images ci-dessous :

```
config: {
    Receptionist: {
        hmm: "../config/model/en-us",
        dict: '../misc/receptionist/asr.dict',
        kwlist: '../misc/receptionist/asr.kwlist',
        gram: "../misc/receptionist/asr",
        grammar: "asr",
        rule: "rule",
        database: {
            "names": "../misc/receptionist/corpus_names.txt",
            "age": "../misc/receptionist/corpus_age.txt",
            "drinks": "../misc/receptionist/corpus_boisson.txt",
            "confirmation": "../misc/receptionist/corpus_confirmation.txt",
            "next": "../misc/receptionist/corpus_next.txt"
        }
    },
    Clean_up: {
        hmm: "../config/model/en-us",
        dict: '../misc/cleanup/cleanup.dict',
        kwlist: '../misc/cleanup/cleanup.kwlist',
        gram: "../misc/cleanup/cleanup",
        grammar: "cleanup",
        rule: "cleanupRule",
        database: {
            "room": "../misc/cleanup/room.txt",
            "confirmation": "../misc/cleanup/confirmation.txt"
        }
    },
    New_Scenario_name: {
        hmm: "../config/model/en-us",
        dict: '../misc/new_scenario_name/new_scenario_name.dict',
        kwlist: '../misc/new_scenario_name/new_scenario_name.kwlist',
        gram: "../misc/new_scenario_name/new_scenario_name_gram",
        grammar: "new_scenario_name_gram",
        rule: "new_scenario_name_gram_rule",
        database: {
            "new_context_name": "../misc/new_scenario_name/verification_file_name.txt"
        }
    }
}
```

```
parser_action_database: {
    Receptionist: {
        askAge: 'age',
        askDrink: 'drinks',
        askName: 'names',
        confirm: 'confirmation',
        askOpenDoor: "next"
    },
    Clean_up: {
        askRoom: 'room',
        confirm: 'confirmation'
    },
    New_Scenario_name: {
        action_name_from_step: 'new_context_name'
    }
}
```

3 – SpeechSynthesisManager :

Rôle : Exécuter la commande de synthèse vocale.

Communications :

- ROS Action avec le HRIManager pour utiliser la commande de synthèse vocale (Serveur : SpeechSynthesisManager, Client : HRIManager)

Fonctionnalités implémentées :

- Synthèse vocale et prononciation d'un discours fourni en paramètre

Fonctionnalités non implémentées :

- Configuration de Mimic dans le fichier de configuration

Configuration :

- A l'heure actuelle, voir https://mycroft-ai.gitbook.io/docs/mycroft-technologies/mimic-overview

4 – connectionManager :

Rôle : Tester la connexion Internet du robot en temps réel et renvoyer l'information au HRIManager

Communications :

- ROS topic publisher pour envoyer l'état de connexion internet du robot (connecté / déconnecté)

Fonctionnalités implémentées :

- Test de la connexion Internet
- Renvoi de l'information au HRIManager