Rapport MGP-I15

Synthèse vocale pour un coach robotique.

Josselin FAYARD, Tom VEILLARD

Février 2016

Table des matières

1 Contexte de l'étude 3

1.1 Le projet de coach (général) 3

1.2 Le robot Poppy 3

2 Cahier des charges fonctionnel 4

2.1 Objectifs 4

2.2 Description Techniques 4

2.3 Délais et livrables 5

3 Description des développements techniques 5

3.1 MaryTTS 5

3.1.1 Description de l'outil 5

3.1.2 Développement de notre solution 6

3.2 Odroid XU4 8

4 Démarche utilisée 10

4.1 Rencontre avec le client pour connaître le sujet : 10

4.2 Etude de la solution : 10

5 Identification des rôles de chaque membre du groupe 12

6 Bilan critique du travail et des résultats. 13

1 Contexte de l'étude

1.1 Le projet de coach (général)

Notre travail s'inscrit dans un projet mené par le département informatique de Télécom Bretagne. Le but de ce projet est de développer un coach robotique humanoïde qui serait capable d'assister les kinésithérapeutes en proposant une supervision des exercices qui doivent être effectués par les patients. Ainsi le robot serait capable de faire une démonstration de l'exercice puis de vérifier que le patient effectue les bons mouvements. Le robot pourrait donc interagir avec le patient pour l'aider à rectifier un mouvement ou par exemple pour l'encourager.

1.2 Le robot Poppy

Pour réaliser toutes ces tâches, c'est le robot Poppy qui a été choisi. Il a été conçu par l'INRIA de Bordeaux, le FlowerLab et l'école de ENSTA ParisTech. C'est donc un robot français qui a la particularité d'être basé sur l'impression 3D ce qui le rend relativement abordable à l'achat par rapport aux autres robots humanoïdes disponibles sur le marché.

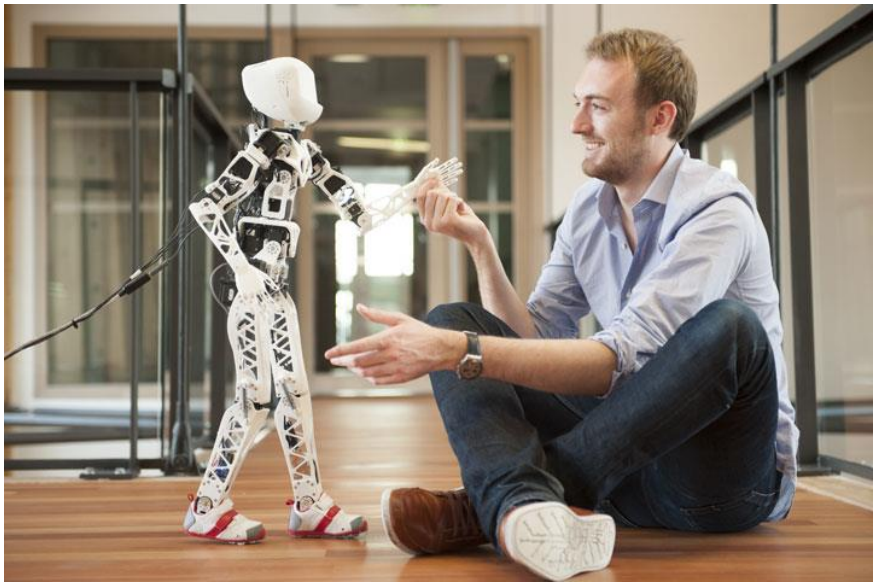


Figure 1: Le robot Poppy

Le robot Poppy est aussi OpenSource et programmable en Python via un framework développé par le FlowersLab. Ce framework permet de programmer facilement en Python des robots basés sur des moteurs Dynamixel, ce qui est le cas de Poppy.

2 Cahier des charges fonctionnel

2.1 Objectifs

Notre projet correspond donc à une brique du coach robotique présenté précédemment. La partie qui nous intéressera est celle qui va gérer la synthèse vocale. Nous devons donc mettre en œuvre un programme qui pourra être utilisé pour gérer la synthèse vocale dans le projet de coach robotique.

2.2 Description Techniques

Pour la réalisation de ce projet, nous devons respecter certaines contraintes pour respecter le besoin initial. Tout d'abord, l'application que nous utiliserons pour la synthèse vocale est MaryTTS (Pour Text To Speech). Ce logiciel a été choisi pour plusieurs raisons. Tout d'abord, c'est un logiciel libre distribué sous licence LGPL[1] ce qui permet d'éviter tout problème juridique et d'un autre de respecter la politique d'utilisation de logiciels libre de Telecom Bretagne. Enfin, ce logiciel ne nécessite pas de connexion à internet pour pouvoir fonctionner et correspond d'autant plus au besoin d'un logiciel embarqué.

La seconde contrainte est que notre programme doit pouvoir fonctionner sur une carte ODROID-XU4. Cette carte est un ordinateur miniature qui sera embarqué dans le robot Poppy. Sur cette carte, c'est un Ubuntu 14.04 LTS qui sera installé car c'est ce système qui est utilisé pour le programme qui contrôlera le robot dans la version complète du projet.

La dernière contrainte que nous devons respecter est que notre programme doit pouvoir s'interfacer avec le robot qui lui, fonctionne en langage Python. Il faudra donc que notre application puisse être appelée facilement via ce langage. Le temps de réponse devra être de l'ordre de 50ms.

[1] La licence est très semblable à la licence GPL qui est plus répandue. Cependant, un logiciel qui utilise des composants sous licence LGPL ne doit pas forcément être lui-même sous licence (L)GPL. Il n'est tout de même pas possible de modifier le code source d'un logiciel sous licence LGPL sans le publier sous licence LGPL. Il est donc possible de choisir da licence la plus appropriée tant que le code source est utilisé tel quel dans le projet.

2.3 Délais et livrables

Du point de vue du projet en lui-même, une version fonctionnelle qui respecte le cahier des charges devra être remise au plus tard en semaine le 11/02/2016, date de la soutenance finale du projet. Un rapport sera remis me 08/02/2016 pour décrire ce projet, autant d'un point de vue technique que d'un point de vue organisationnel.

Nous avons aussi prévu de faire un minimum d'un point d'avancement par semaine avec Mai Nguyen, qui détient le rôle de maitrise d'ouvrage du projet.

3 Description des développements techniques

3.1 MaryTTS

3.1.1 Description de l'outil

MaryTTS est donc un logiciel de synthèse vocale open sources et multi-langues. Son fonctionnement est basé sur un serveur qui sera en local sur la carte ODROID-XU4 et qui peut être appelé de plusieurs manières différents pour générer un fichier audio à partir d'un texte brut.

On peut donc prévoir certaines limitations. Par exemple, il sera impossible pour MaryTTS de générer le fichier audio si la langue du texte n'est pas la même que la langue utilisée pour la synthèse vocale.

Après quelques tests, nous avons remarqué d'autres limitations comme une prononciation approximative de certains mots ou même une confusion entre plusieurs mots au sens différents. Par exemple le verbe être conjugué au présent de l'indicatif à la troisième personne du singulier sera interprété comme étant le verbe « être » conjugué à la troisième personne du singulier mais à l'imparfait. Ce genre de défaut devra donc être prévu en privilégiant l'utilisation de mots dont la prononciation est mieux gérée ou en utilisation des mot phonétiquement semblables à ceux qui peuvent être mal interprétés.

3.1.2 Développement de notre solution

Pour le développement de notre solution, nous avons choisi de définir plusieurs lots de manière à mieux pouvoir anticiper un éventuel retard sur notre planning initial.

3.1.2.1 LOT 1 : « Proof of concept » de l'utilisation de MaryTTS

Pour ce premier lot, nous avons choisi de développer un programme capable d'utiliser un serveur MaryTTS pour générer un fichier audio et de le lire.

Nous avons choisi de le développer en langage Java, car nous avons déjà cette compétence au sein du groupe et que l'apprentissage d'un nouveau langage pourrait amener induire des risques supplémentaires liés à une mauvaise utilisation de ce nouveau langage ou à une vitesse de développement trop lent par rapport au planning que nous avons fixé.

3.1.2.2 LOT 2 : Ajout des phrases en plusieurs langues dans des fichiers séparés

La seconde partie du travail a donc consisté définir des phrases types qui seront utilisée par le robot pour les stocker dans des fichiers XML qui seront lus par notre application. Nous avons tout d'abord choisi d'utiliser des phrases construites de la manière suivante :

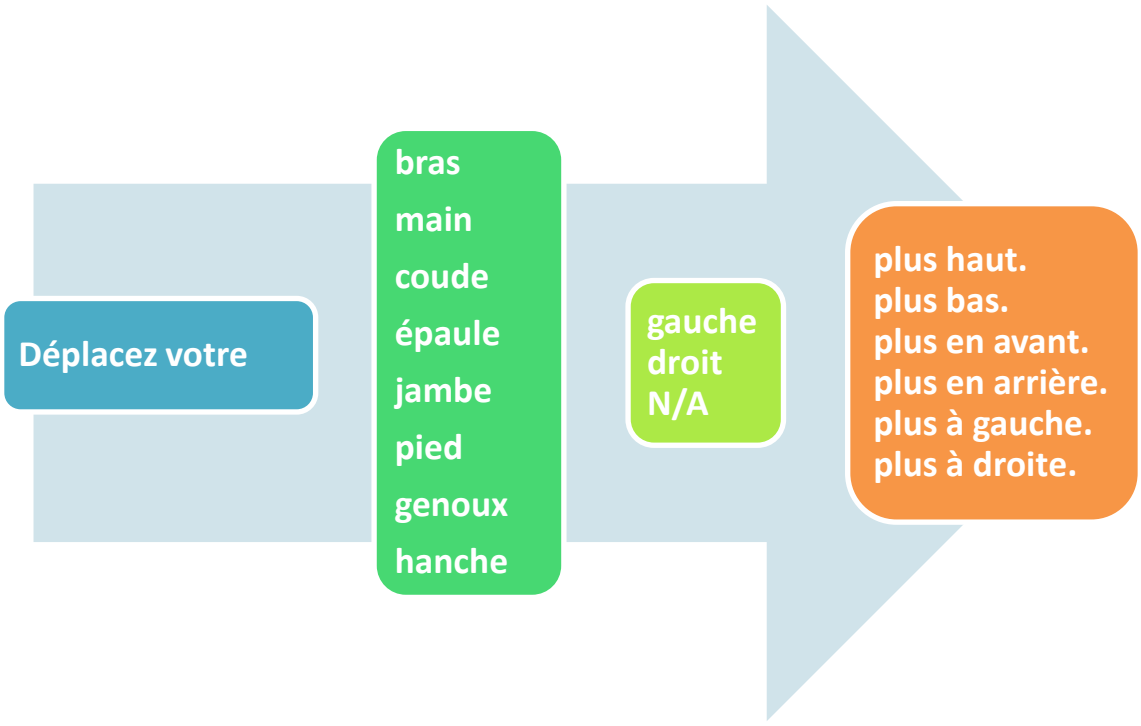


Figure 2: Construction de phrases à prononcer

Une phrase type pourra donc être : *« Déplacez votre bras droit plus haut. »*

Dans le fichier XML, le texte est formaté comme suit :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<resources>
<string name="foot-left-back">Déplacez votre pied gauche plus en arrière</string>
<string name="foot-left-left">Déplacez votre pied gauche plus à votre gauche</string>
<string name="foot-left-right">Déplacez votre pied gauche plus à votre droite</string>
<string name="foot-right-up">Déplacez votre pied droit plus haut</string>
……
```

Grace à ce système, on peut facilement retrouver les phrases voulues grâce à trois mots clés qui sont la partie du corps concerné, le côté et la direction du mouvement.

Le programme peut donc maintenant être lancé avec des arguments qui décrivent les trois mots clés ci-dessus.

3.1.2.3 LOT 3 : Mise en place d'une architecture client-serveur

Dans les contraintes listées plus tôt, l'une d'entre elles était le temps de réponse qui doit être de l'ordre de 50ms. Le problème avec le programme précédent est que la machine virtuelle Java doit se lancer à chaque fois qu'une nouvelle phrase doit être prononcée. Pour le résoudre, nous avons mis en place une architecture client/serveur.

Un serveur est donc lancé sur le ODROID-XU4 en plus du serveur MaryTTS. Ce serveur accepte donc des connexions TCP sur un port particulier pour recevoir une chaine de caractères qui décrit les options à utiliser pour la synthèse vocale. Ainsi, on accélère grandement le fonctionnement du programme.

Pour se connecter au serveur, nous avons développé un script en langage Python qui pourra être ajouté au programme du robot.

3.2 Odroid XU4

Pour réaliser la solution technique nous disposions d'un Odroid XU4, c'est la dernière génération d'ordinateurs mono-cartes développés par Hardkernel (Entreprise coréenne), disposant de 3 ports USB (2xUSB 3.0 / 1xUSB 2.0) d'une entrée Ethernet ainsi que de deux types de stockage (eMMC5 et microSD).

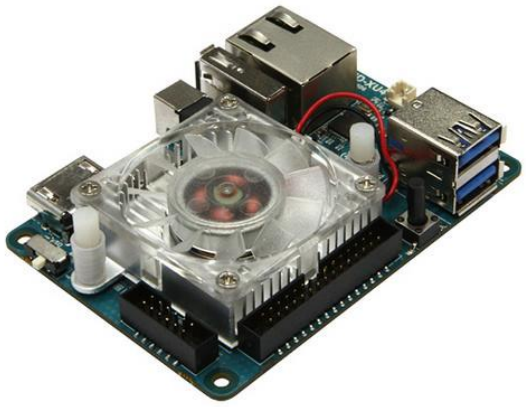


Figure 3 : Ordroid UX4

La première étape a été de charger la version de Ubuntu 14.04.01 imposé par le client sur la carte eMMC5. Une fois l'OS récupéré nous avons préparé le eMMC module en le flashant, puis en lui injectant le nouvel OS via l'adaptateur eMMC-SD/MMC.

Pour le 1er démarrage de l'odroid, nous avons réalisé un réseau local entre un ordinateur et l'odroid via un câble Ethernet. Par défaut, l'odroid fait une demande de DHCP discover pour obtenir une adresse IP, nous avons donc utilisé le logiciel openDHCP pour installer un service DHCP sur notre l'ordinateur afin qu'il attribue une adresse à l'Odroid. Une fois l'adresse IP configuré nous avons réalisé une connexion SSH pour accéder à l'Odroid.

Nous avons ajouté un dongle Wi-Fi que nous avons configuré de façon à ce que l'odroid puisse accéder à internet. De ce fait nous avons pu télécharger toutes les librairies nécessaires au fonctionnement du Framework MaryTTS.

Inventaire :

Nous avions besoin d'une entré USB pour l'adaptateur USB/jack, d'une entrée USB pour l'alimentation des enceintes externes ainsi qu'un USB pour le dongle Wi-Fi, heureusement l'odroid offre 3 ports USB.

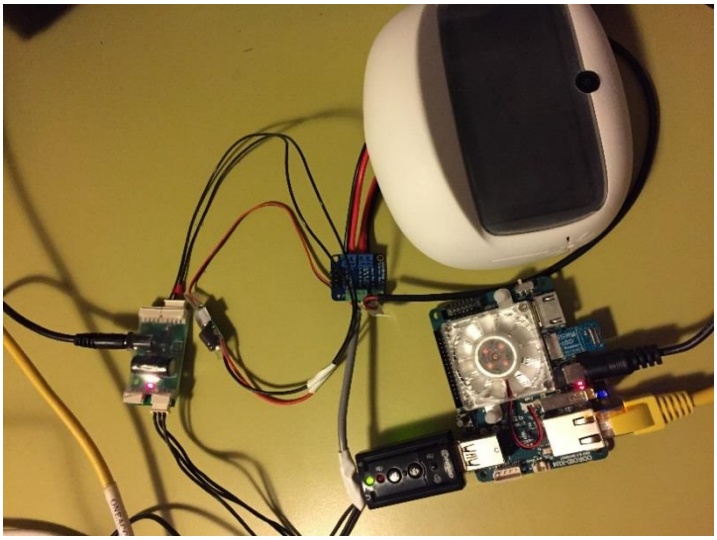


Figure 4 : Odroid rattaché au robot Poppy ainsi que des modules nécessaires à son utilisation

Nous avons en premier lieu testé le framework MaryTTS avec le client/serveur proposé par le site. Afin d'obtenir l'interface graphique du client, nous avons déployé un serveur d'affichage (Ming) et redirigé l'affichage de l'odroid (via une connexion sh putt) sur nos ordinateurs personnels. Une fois l'environnement de bureau disponible nous avons pu tester le fonctionnement de MaryTTS sur l'odroid.

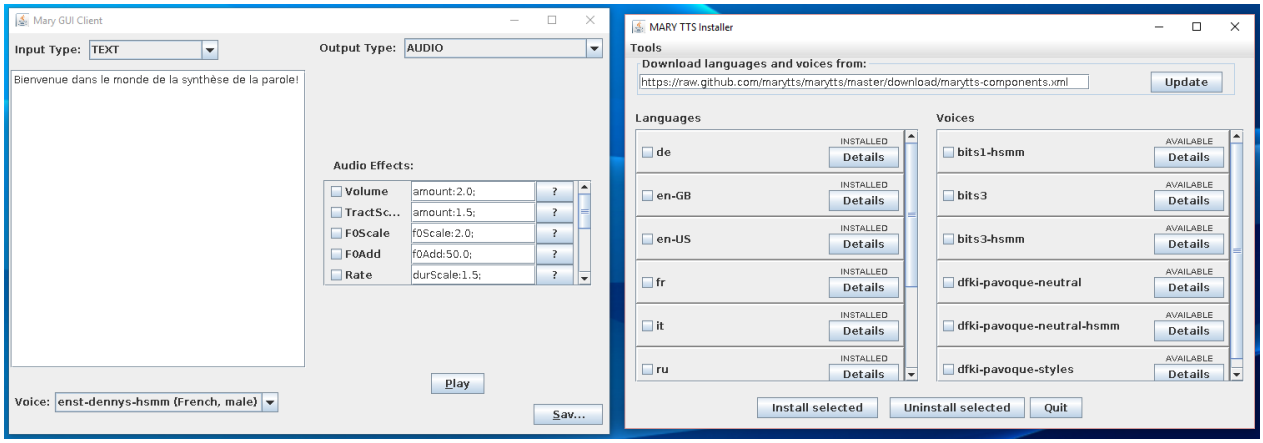


Figure 5: Client et serveur de MaryTTS

Durant le déploiement, nous avons rencontré un certain nombre de problèmes notamment sur le fonctionnement des enceintes présentent sur la tête du robot, ainsi que sur des temps de latence élevées entre l'envoi d'un texte et sa lecture audio. Cependant, l'ensemble de ces problèmes ont pu être résolus au fils du projet.

4 Démarche utilisée

Pour réaliser ce projet nous avons suivi un certain nombre de démarches :

4.1 Rencontre avec le client pour connaître le sujet :

Cette 1ere phase nous a permis de rencontrer notre tuteur (qui est également le client) pour ce projet. Le but de cette rencontre était de clarifier l'idée du projet de synthèse vocale. Nous avons donc fait un point avec notre client pour déterminer le concept général du projet, puis nous avons pu mettre au clair les objectifs attendus pour la fin du projet et fait un point sur les besoins matériels que nous allions utiliser.

4.2 Etude de la solution :

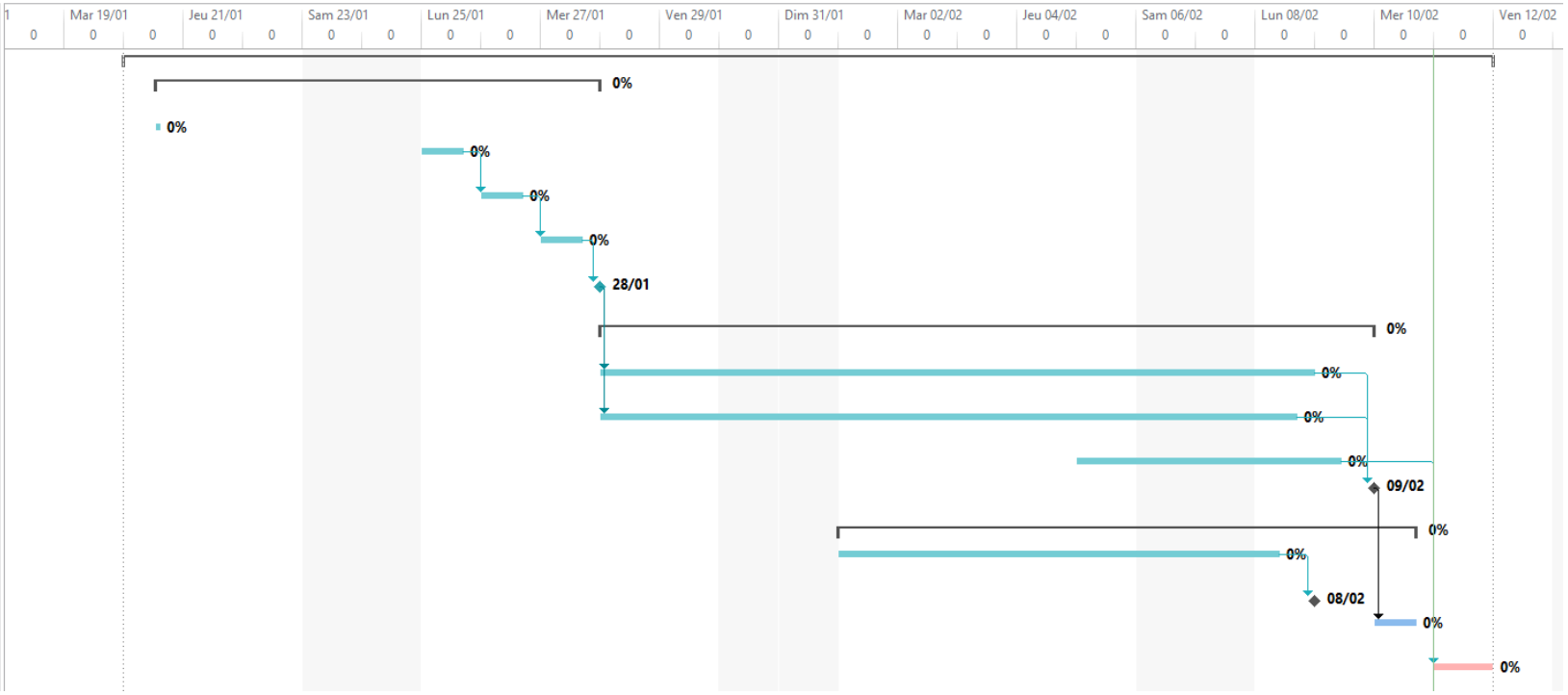
En seconde partie, nous avons étudié la solution que nous allions déployer, pour ce faire nous avons analysé les documentations liées au Framework MaryTTS ainsi qu'à l'Odroid.

Nous avons réfléchi sur les différents livrables que nous devions réaliser :

- Cahier des charges
- 1er recette qui pour présenter l'avancement du projet
- Rapport final
- Soutenance et recette finale du projet

Suite à cela nous nous sommes réparti les tâches et réalisé un Gantt prévisionnel pour le projet :

WE	Nom de la tâche	Durée
1	**Synthèse vocale**	**15 jours**
1.1	**Phase 1 : Découverte du projet**	**4,88 jours**
1.1.1	Définition du projet	2 h
1.1.2	Lecture de la documentation	7 h
1.1.3	Etude de la solution technique	7 h
1.1.4	Mise en place du cahier des charges	7 h
1.1.5	Validation du cahier des charges par le client	0 jour
1.2	**Phase 2 : Déploiement technique**	**7,88 jours**
1.2.1	Developpement de la solution MarryTTS	7 jours
1.2.2	Mise en place du micro ordinateur Odroid UX4	7 jours
1.2.3	Test de la solution	2 jours
1.2.4	Validation de la solution avec le client	0 jour
1.3	**Phase 3 : Documentation**	**7 jours**
1.3.1	Réalisation de la documentation	4,5 jours
1.3.2	Remise du rapport	0 jour
1.3.3	Préparation de la soutenance	7 h
1.4	Soutenance	7 h



Nous avons découpé le projet en 3 phases chacune délimité par un livrable :

La première phase qui correspond à la découverte du projet s'est conclue par la réalisation d'un cahier des charges validant les différents besoins du client.

La seconde phase a été la phase dite « technique » pendant laquelle nous avons développé et déployé une solution pour le projet. Cette phase technique s'est conclue par la présentation de la solution technique à notre client.

Enfin, la dernière phase est celle de la documentation, qui regroupe l'ensemble des documents nécessaires pour comprendre ce que l'on a réalisé et permettre à une personne de poursuivre le projet. Cette dernière partie c'est achevé par la rédaction d'un rapport, ainsi que de l'ajout dans un wiki des informations nécessaire au fonctionnement et la compréhension de notre solution. Pour réaliser ce projet nous disposions de plusieurs moyens techniques facilitant le travail en groupe.

Au niveau du développement de l'outil, nous avons opté pour l'utilisation du logiciel de gestion de versions décentralisé Git. Pour ce qui est des différents documents et livrables nous avons utilisé l'outil « google doc » permettant aux deux membres du groupe d'avoir accès aux travaux de chaque personne à tout moment.

L'utilisation de nos propres outils au lieu de se proposer par l'école vient du fait que la durée et le nombre de personnes dans l'équipe étaient réduit, ainsi pour un gain de temps nous avons préféré utiliser des outils que nous maitrisions déjà.

5 Identification des rôles de chaque membre du groupe

Lors de ce projet, nous avons partagé les tâches en fonction de l'avancement du projet. Pour la partie de développement du programme en Java a été gérée par Tom Veillard et l'implémentation sur la carte ODROID-XU4 ainsi que le développement en Python. A été gérée par Josselin Fayard. Nous avons aussi divisé les parties de rédaction et d'étude préliminaire en deux pour pouvoir travailler de façon plus efficace.

6 Bilan critique du travail et des résultats.

Le déroulement du projet c'étant bien passé, nous avons pour le moment bien respecté notre planning prévisionnel, même pris une journée d'avance sur la livraison du produit. Durant la réalisation du projet, nous avons réalisé plusieurs points avec notre client de manière à être en accord avec ce qu'il souhaitait comme résultat final. Nous avons réussi à valider les points majeurs requis par le cahier des charges.

Nous avons finalement développé un programme java utilisant le framework MaryTTS pour répondre aux besoins de la problématique de « text to speech ». Afin d'interférer avec les autres programmes qu'utilisera Poppy, ils nous avaient été demandé de réaliser une interface pour l'utilisateur en Python. Pour réaliser cette interface nous avons développé un script en Python s'interfaçant avec notre application via l'utilisation d'un modèle client/serveur.

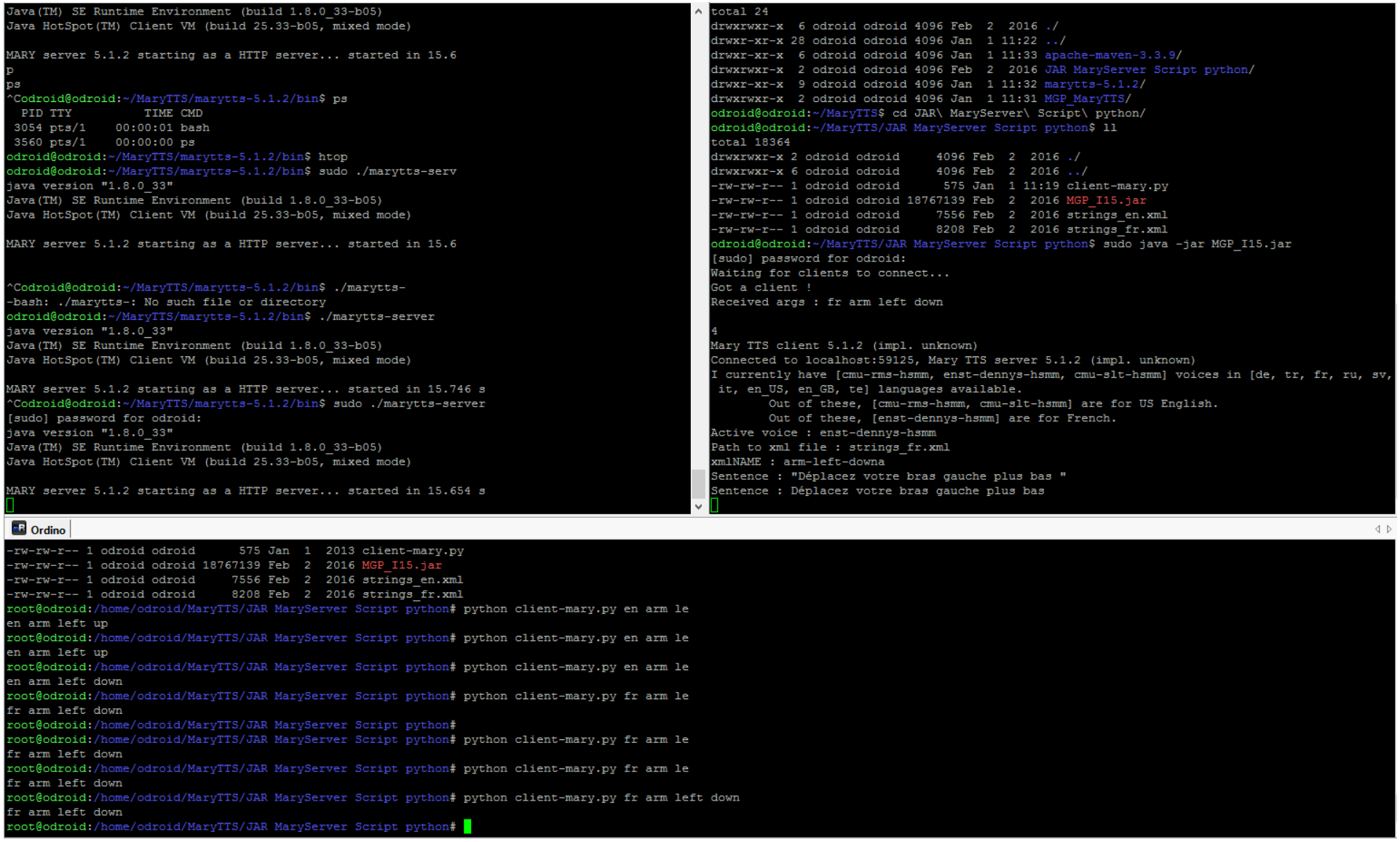
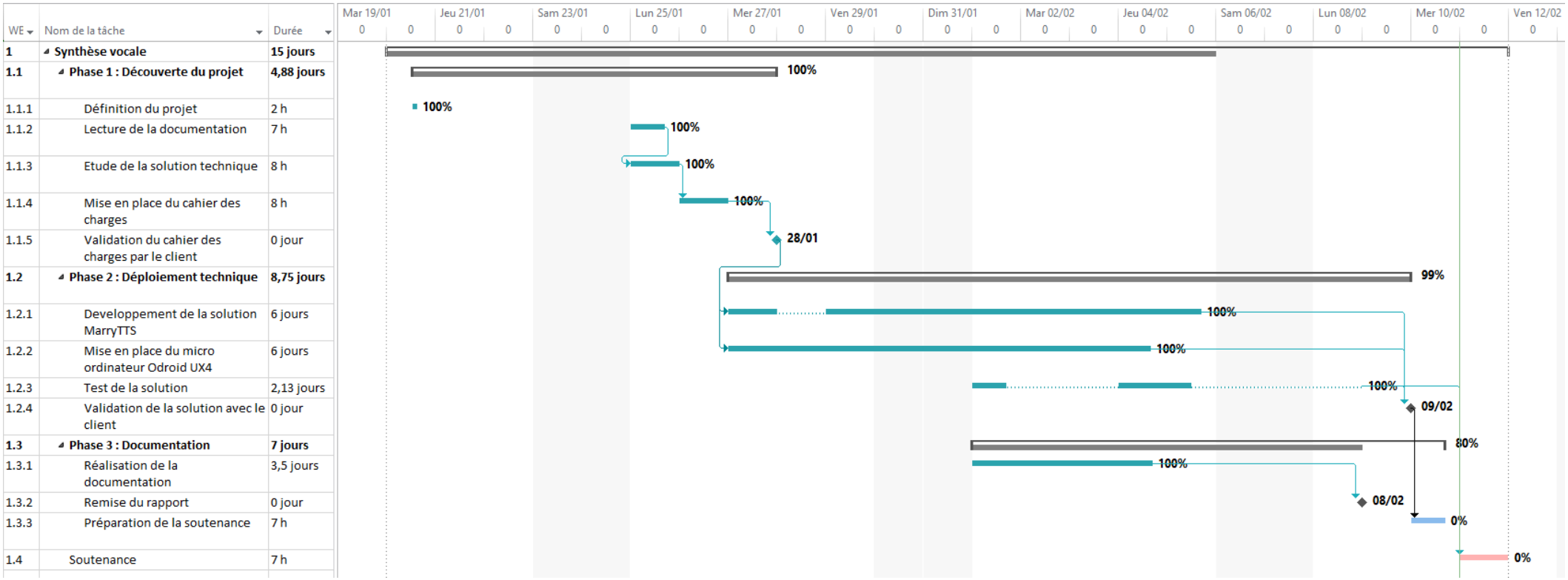


Figure 6 : Server maryTTS (écran en haut à gauche), Service développé en Java qui attend des requêtes générées à partir d'un script Python (en haut à droite), Exemple d'utilisation sur script Python (écran en bas)

En matière de méthode de travail, nous pensons que pour un projet de cette envergure (2 semaines avec 2 collaborateurs), l'utilisation des outils de gestion de projet (Gantt, redmine…) était un peu démesurée. Nous avions, un peu surestimé la partie étude du projet (qui représente environ 40% du temps de travail), en effet, dans notre cas de figure beaucoup de documentation était présente pour nous aider à utiliser le Framework ainsi que l'odroid. Ainsi dans nos travaux effectifs, nous avons pris de l'avance sur le déroulement général du projet.



WE	Nom de la tâche	Durée
1	**Synthèse vocale**	**15 jours**
1.1	**Phase 1 : Découverte du projet**	**4,88 jours**
1.1.1	Définition du projet	2 h
1.1.2	Lecture de la documentation	7 h
1.1.3	Etude de la solution technique	8 h
1.1.4	Mise en place du cahier des charges	8 h
1.1.5	Validation du cahier des charges par le client	0 jour
1.2	**Phase 2 : Déploiement technique**	**8,75 jours**
1.2.1	Developpement de la solution MarryTTS	6 jours
1.2.2	Mise en place du micro ordinateur Odroid UX4	6 jours
1.2.3	Test de la solution	2,13 jours
1.2.4	Validation de la solution avec le client	0 jour
1.3	**Phase 3 : Documentation**	**7 jours**
1.3.1	Réalisation de la documentation	3,5 jours
1.3.2	Remise du rapport	0 jour
1.3.3	Préparation de la soutenance	7 h
1.4	Soutenance	7 h

Comme indiqué sur le Gantt, les parties de lecture et étude de la solution se sont avérées être plus rapide. Nous nous sommes donc permis de débuter l'élaboration du prototype avant même que le cahier des charges ne soit finalisé. Cette journée d'avance nous a permis d'entrevoir un peu l'utilisation du Framework MaryTTS ainsi que de l'installation de l'Odroid XU4.