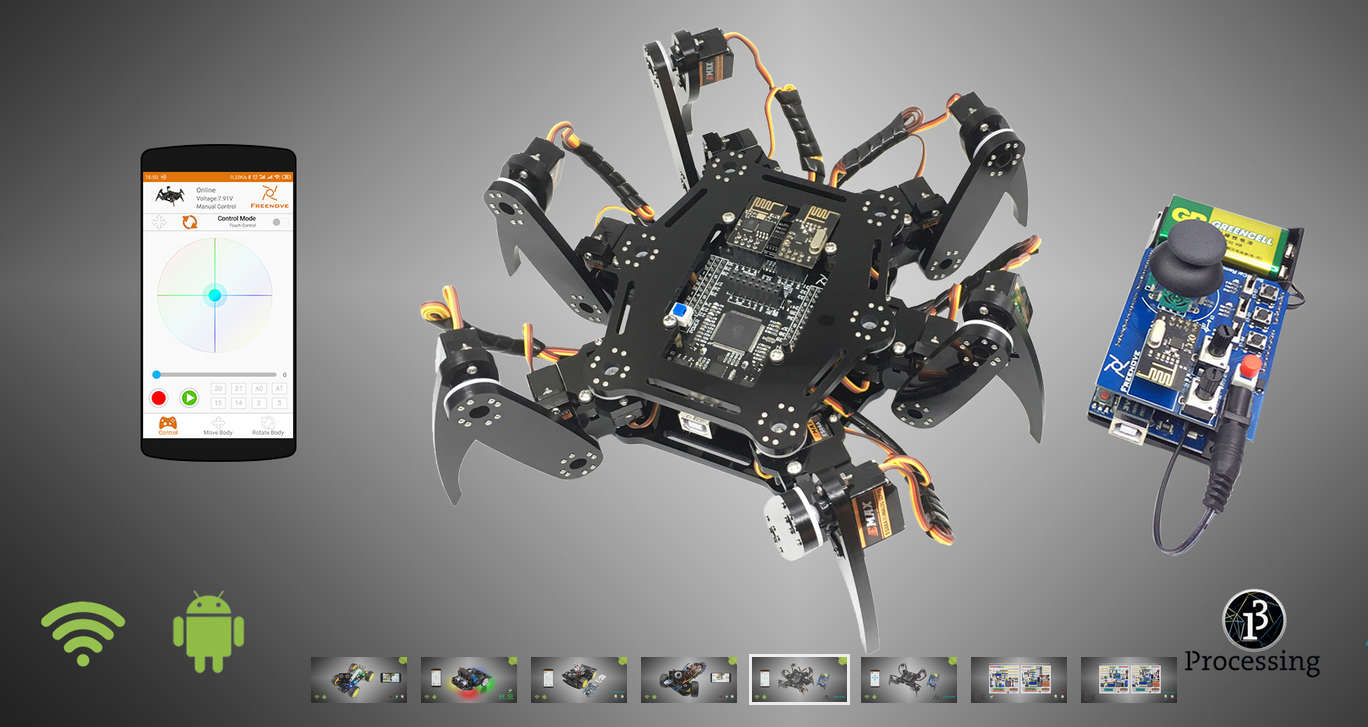
LP IOTIA-SIGD Année 2019-2020

Rapport de projet Metal-Gear

Contrôle Hexapod

*Projet Réalisé par*

Cordier Florent

Hervé Louis

Stéphane Azoulay

et Flavien jalbert

*Projet encadré par*

Peraldi Frati Marie-Agnès

Bruneau Jean-Michel

et Cécile Belleudy

# Sommaire

## Introduction

## Cahier des charges

Fonctionnalités du projet

Répartition des tâches

Architecture matérielle

Architecture logicielle

Matériel

Scénarios d’usages

Fonctionnalité avec les codes

# Bilan

Ce qui a été fait

Ce qu’il reste à faire

# Lexique

Introduction

Le défi de ce projet est simple : contrôler un robot par la voix.  
On distingue donc 3 informations dans cet énoncé. Premièrement la notion de contrôle, le terme de robot et enfin l’utilisation de la voix.  
Nous avions aussi quelques contraintes telles que l’utilisation d’un protocole sans fil comme le Bluetooth ou le WiFi pour ne citer que ceux là, la mise en place d’un système d'agrégation d’échange comme OneM2M ou MQTT ainsi qu’une partie web et/ou mobile.  
  
Nous nous sommes donc posé la question : comment intégrer toutes ces contraintes dans notre défi ?

Et c’est ainsi que nous avons eu l’idée d’utiliser tous les éléments qui vont suivre et être détaillés dans le Cahier des Charges.

Cahier des charges

# Fonctionnalités du projet

* **F1 Pilotage de Freenove© Hexapod :**

**F1.1** Pilotage du Freenove Hexapod par WiFi

**F1.2** Retranscrire les instructions vocale vers le micro service web (API) de coordination

|  |  |  |  |
| --- | --- | --- | --- |
| Code Hexapod | Variable | Commande Vocale | API Endpoint |
| 0 | requestEcho |  | /ping/hexapod |
| 1 | echo |  |  |
| 10 | requestSupplyVoltage |  | /ping/voltage |
| 11 | supplyVoltage |  |  |
| 20 | requestChangeIO |  |  |
| 30 | requestMoveLeg |  |  |
| 32 | requestCalibrate | calibre-toi | /action/calibrate |
| 64 | requestInstallState |  |  |
| 66 | requestCalibrateState |  |  |
| 68 | requestBootState |  |  |
| 70 | requestCalibrateVerify |  |  |
| 80 | requestCrawlForward | avance | /movement/forward/:qty |
| 82 | requestCrawlBackward | recule | /movement/backward/:qty |
| 84 | requestCrawlLeft | rampe à gauche | /movement/left/:qty |
| 86 | requestCrawlRight | rampe à droite | /movement/right/:qty |
| 88 | requestTurnLeft | tourne à gauche | /turn/left/:qty |
| 90 | requestTurnRight | tourne à droite | /turn/right/:qty |
| 92 | requestActiveMode | allume-toi | /action/wakeup |
| 94 | requestSleepMode | éteins-toi | /action/standby |
| 96 | requestSwitchMode |  |  |
| 110 | requestCrawl | rampe | /movement/:qty |
| 112 | requestChangeBodyHeight | change ta hauteur | /height/:height |
| 114 | requestMoveBody | bouge ton corps | /move/:qty |
| 116 | requestRotateBody | tourne-toi | /rotate/:qty |
| 118 | requestTwistBody | dance | /twist/:qty |
| 21 | orderStart |  |  |
| 23 | orderDone |  |  |
| 128 | transStart |  |  |
| 129 | transEnd |  |  |
|  |  |  |  |

**F1.3** Retranscrire les instructions http vers la CLI (Un micro service web (API) de coordination)

* **F2 Log des déplacements de Freenove© Hexapode**

**F2.1** logs des informations (type de contrôle, origine, date, quantité de mouvement, exécution de l’ordre, ordre effectué, erreur reçue (optionnel), description de l’erreur (optionnel))

**F2.2** Création de points d’accès de l’historique des instructions de déplacement du Freenove Hexapod

* **F3 Une application web et Mobile pour contrôler la Freenove© Hexapod à distance**

**F3.1** Pilotage à distance via des commandes envoyées en http par une application web / mobile

**F3.2** Envoyer des instructions de déplacement

**F3.3** Recevoir et afficher le journal d’événement

* **F4 Contrôle sécurisé via https**

**F4.1** Communication https via internet

**F4.2** Générer et distribuer des certificats

**F4.3** Sécuriser le réseau WiFi du Freenove Hexapod

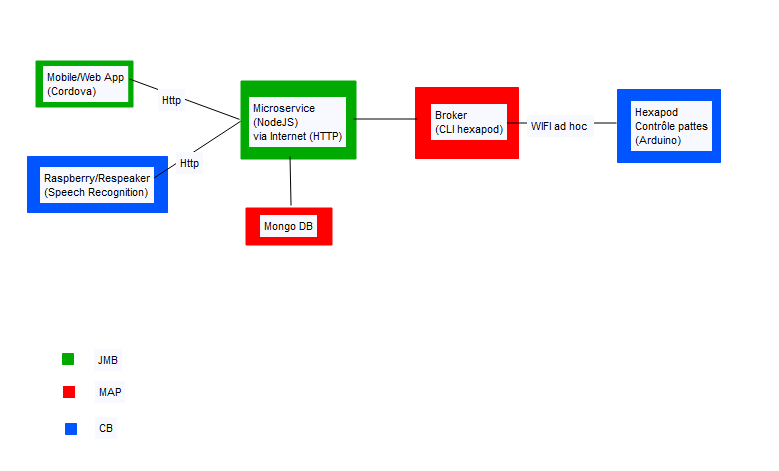
# Répartition des tâches

* **Louis Hervé :** F1.1, F1.2
* **Flavien Jalabert :** Montage du Freenove© Hexapod, F1.3, F2.2, F3, F2.1
* **Florent Cordier :** Montage du Freenove© Hexapod, F3, F4.3
* **Stephane Azoulay :** F1.2, F4

# Architecture matérielle

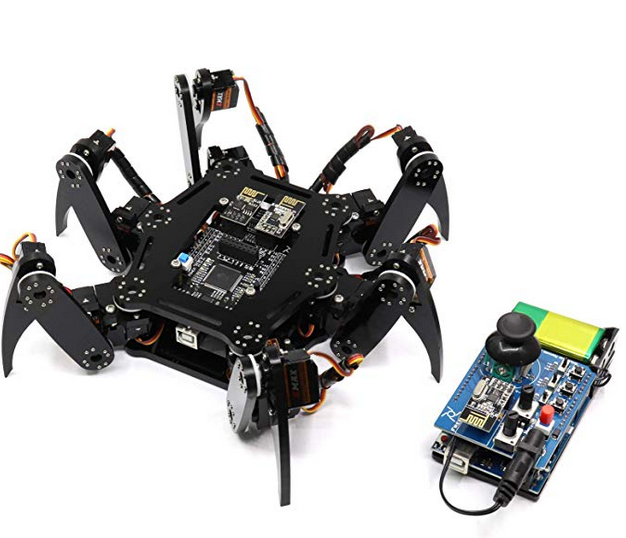
# 

# Architecture logicielle



# Matériel

Hexapod Freenove

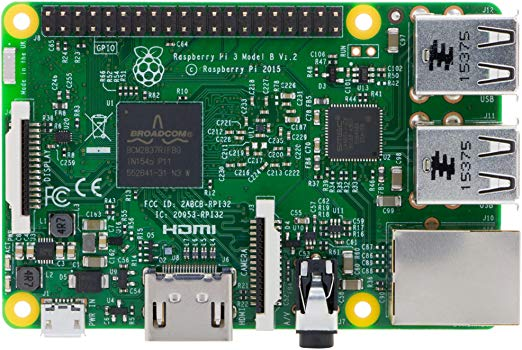


Raspberry Pi 3

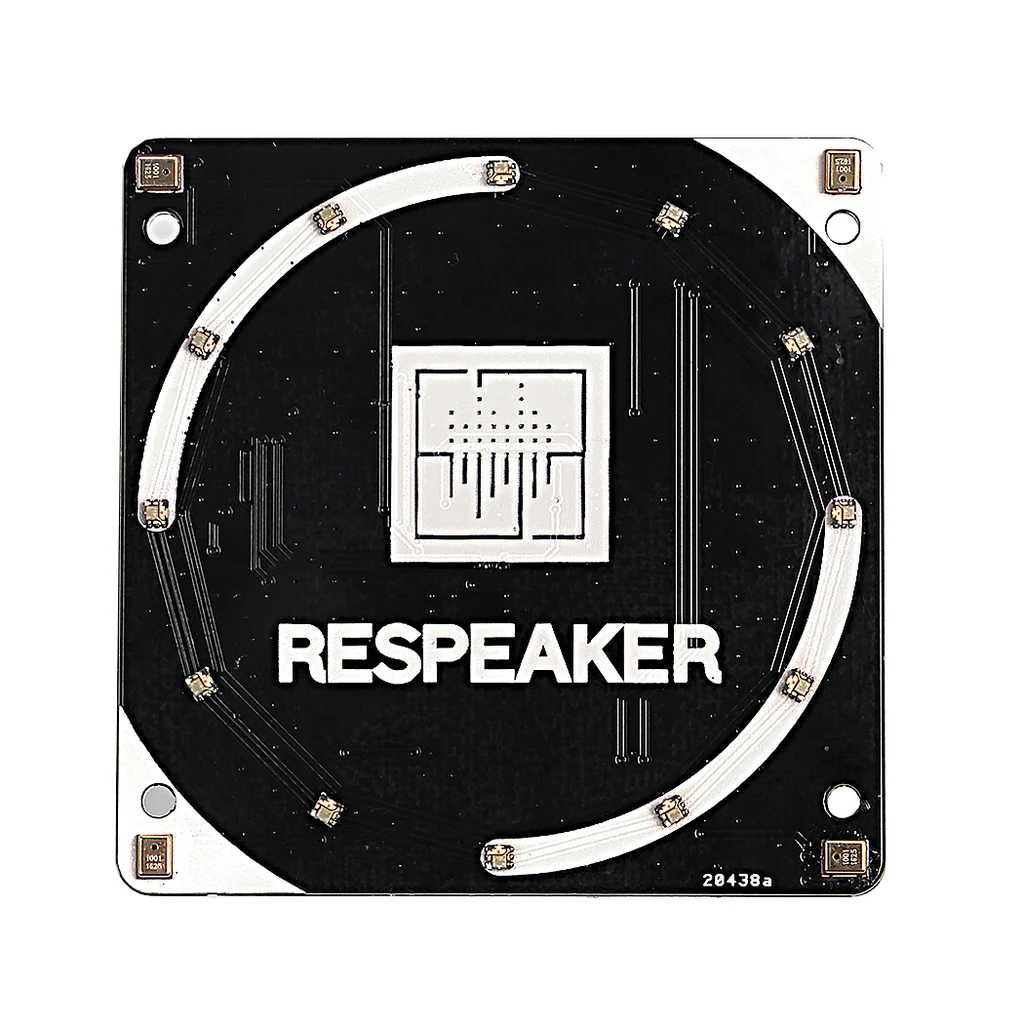
x1 Câble HDMI

x2 USB pour le clavier et la souris

x1 Câble USB pour l’alimentation de l’appareil



Respeaker 4-Mic Array pour Raspberry Pi



# Scénarios d’usage

**S1 :** Contrôle par la voix du Freenove© Hexapod

* **ReSpeaker :** Les commandes vocales sont traitées sur la raspberry. Le CLI python convertit les commandes vocales en instructions à envoyer au Freenove© hexapod : il génère une requête http pour le micro service de coordination qui va ensuite communiquer avec la CLI qui va via WiFi envoyer l’instruction traitée (en fonction du besoin) au module wifi esp8266 du Freenove hexapod.

**S2 :** Pilotage à distance

* **End devices :** L’application web/mobile va envoyer des instructions pour la Freenove© Hexapod en passant par l’API du micro service de coordination..Ensuite, la CLI enverra les instructions traitées au Freenove© Hexapod.

**S3 :** Journal d’événement

* **MongoDB:** L’api est montée avec une base de donnée mongodb qui se chargera de conserver toutes les instructions reçues/transmises entre le l’api et la CLI sous forme de journal d’événements

**S4 :** Accès á l’historique

* **End Devices :** Grâce au journal d’événements ainsi qu’au serveur de coordination, depuis les applications web/mobile, nous pouvons visualiser les instructions reçues par le Freenove© Hexapod en temps réel ou obtenir un historique.

# Réalisation des fonctionnalités + codes associés

CLI hexapod : (Hervé Louis)

dossier [broker](https://github.com/mapfra/LPIOTIA_2019_MetalGear/tree/master/broker)

F1.1 : [lien github](https://github.com/mapfra/LPIOTIA_2019_MetalGear/blob/master/broker/hexapod)

Montage Arduino : (Cordier Florent, Jalabert Flavien)

dossier [arduino](https://github.com/mapfra/LPIOTIA_2019_MetalGear/tree/master/arduino)

API de coordination : (Jalabert Flavien)

dossier [server](https://github.com/mapfra/LPIOTIA_2019_MetalGear/tree/master/server/coordinationServer)

F1.3 : tout le dossier

F2.1 : [lien github](https://github.com/mapfra/LPIOTIA_2019_MetalGear/blob/a98b3f95fe61ff0d9051fccb5a59560bc84bd45a/server/coordinationServer/modules/services/orders/server/controllers/orders.server.controller.js#L77)

F2.2 : [lien github](https://github.com/mapfra/LPIOTIA_2019_MetalGear/blob/master/server/coordinationServer/modules/services/eventLogger/server/controllers/events.server.controller.js)

Respeaker : (Hervé Louis, Azoulay Stéphane)

dossier [voiceControl](https://github.com/mapfra/LPIOTIA_2019_MetalGear/tree/master/voiceControl/Python)

F1.2 : [lien github](https://github.com/mapfra/LPIOTIA_2019_MetalGear/blob/master/voiceControl/Python/voice_recognition.py)

Application web/mobile : (Cordier Florent, Jalabert Flavien)

dossier [web\_mobile](https://github.com/mapfra/LPIOTIA_2019_MetalGear/tree/master/web_mobile/application)

F3.1 : tout le dossier

F3.2 : [lien github](https://github.com/mapfra/LPIOTIA_2019_MetalGear/blob/a98b3f95fe61ff0d9051fccb5a59560bc84bd45a/web_mobile/application/www/index.html#L34)

F3.3 : NYI

F4 : NYI

Bilan

A la fin du confinement (et entre autre du projet), nous avons réussi à piloter l’hexapod en utilisant notre voix, Le processus par lequel la voix passe avant d’arriver à l’hexapod comprend le traitement de la voix (séparation des commandes et des distances), la requête est alors générée pour le serveur de coordination qui spawn une ligne de commande pour notre custom Command Line Interface qui va envoyer les bytes prévus pour chaque action le nombre de fois nécessaire pour que le robot se déplace de la distance demandée.  
L’application web et mobile a aussi été réalisée, elle permet les mêmes actions que par la voix. Les actions envoyées par l’application passe par le même serveur de coordination et donc suis le même processus.  
  
Dans les fonctionnalités prévues, l’une d’entre elle n’a pas pu voir le jour : la lecture des événements enregistrés par le serveur de coordination ne peuvent pas à ce jour être visualisés par l’application.

De plus, nous avions pensé à quelques possibilités supplémentaires si le temps nous l’avait permis, comme par exemple ajouter une caméra sur le robot qui permettrait de retranscrire sa vision sur l’application grâce à un système de streaming OTT, ou la possibilité de demander à l’hexapod de faire une action complexe qui serait un enchaînement de plusieurs actions basiques (tourner sur soi même, danser, avancer en L comme le cavalier aux échecs, etc…).

Parmi les conseils qui nous ont été donnés, l’un deux avait retenu notre attention. Sécuriser le réseau dans lequel la raspberry et l’hexapod communiquent afin d’éviter qu’une personne extérieure connaissant le robot puisse envoyer des ordres sans que nous ne nous en apercevions.  
Ou encore de monter l’API sur un serveur https avec une autocertification, encore une fois, manque de temps dû à la situation sanitaire mondiale.

Vidéo de démonstration du projet disponible en ligne sur [google drive](https://drive.google.com/file/d/1xvcZtUSglIbRr2Is79H4Op1UhGKgTNxN/view?usp=sharing).

Lexique

**Raspberry PI 3 :**

Le Raspberry PI est un nano-ordinateur monocarte conçu par des professeurs du département informatique de l'université de Cambridge. Il permet l’exécution de plusieurs systèmes d’exploitation. Ici Nous tournerons sur une distribution de Debian appelée Raspbian.

**Respeaker :**

Est une carte d'extension à quadruple microphone pour Raspberry Pi conçue pour les applications IA et voix. Cela signifie que vous pouvez créer un produit vocal plus puissant et plus flexible qui intègre le service Amazon Alexa Voice, l'Assistant Google, etc.

**Freenove© Hexapod :** .

Hexapod est un robot développé par Freenove© compatible avec arduino, il est composé en plus d’un module WiFi esp8266 pour pouvoir le contrôler à distance.

**Module WiFi esp8266 :**

L'ESP8266 est un circuit intégré avec un microcontrôleur permettant la connexion en WiFi. Les modules intégrant ce circuit sont très utilisés pour contrôler des périphériques par Internet.