Projet 13
Abt Juan Francisco, Chauvet Louis
Dadam Federico, Sola Francisco,
Zhang Hang

Rapport technique – Projet INGS4

Main préhensile pour le robot Poppy et interface tangible

À l'attention des encadrants techniques et de Génération Robots

Encadrants : Devanne Maxime, Lohr Christophe, Ménard Jacky, Nguyen Maï, Amigo Isabel, Simonnet Mathieu

Client partenaire industriel : Damien Deguyenne, Génération Robots.

Auteur(s) : Abt Juan Francisco, Chauvet Louis, Dadam Federico, Sola Francisco, Zhang Hang

Version finale - 21 juin 2018
Formation Ingénieur 2A
Année scolaire 2017-2018

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Résumé

Ce projet a pour objectif de permettre au robot Poppy Torso, de la famille des robots Poppy d'interagir avec une table interactive Reactable et un humain.
L'interaction est à la fois logicielle et matérielle (la main et les objets doivent être à la fois manipulable par le robot et détectable par la table). Pour cela il faut fabriquer une main préhensile pour ce robot, qui soit capable de déplacer des objets à l'aide d'un électroaimant lui donnant la capacité de les saisir.

Dans un but de démonstration et de validation avec notre client, nous faisons jouer le robot à un jeu simple avec un humain, comme le tic-tac-toe. Ce jeu aura lieu sur la surface d'une table interactive appelée Reactable.

Afin de bien mener notre travail de recherche, ce groupe de travail s'est appuyé sur l'analyse des travaux précédents dans ce domaine et sur des simulations du comportement du robot.

Ambitieux, le projet s'étale sur différentes parties rassemblant plusieurs disciplines comme la mécanique, l'électronique, la programmation sans oublier la gestion et la planification.

Le présent document a pour but de rendre compte du travail réalisé durant ces cinq derniers mois et des résultats ainsi obtenus afin de permettre à des futurs projets de s'appuyer dessus pour leurs recherches ou de reprendre le travail effectué.

Table des matières

Table des illustrations

Introduction

Le projet qui fait l'objet de ce rapport final a eu lieu durant le premier semestre de l'année 2018 dans le cadre des projets S4 de l'IMT Atlantique.
Le client intervenant dans ce projet est une entreprise appelée Génération Robots, M. Jérôme Laplace étant son CEO et responsable. Cependant, le correspondant avec qui nous échangerons est M. Damien Deguyenne.

L'entreprise se caractérise par son travail dans le monde de la recherche et l'ingénierie appliquée à la robotique. Ses spécialisations sont entre autres la conception et industrialisation de capteurs pour la robotique, la formation à la programmation de robots, programmation des robots de service afin de démontrer ce qui peut être fait en termes de comportement interactif évolué à l'aide de ces robots. Ces développements s'adressent aux laboratoires de recherche ainsi qu'aux professionnels désireux d'incorporer dans leur activité les dernières avancées de la robotique de service professionnelle.

D'un côté nous étions encadrés par deux enseignants/chercheurs de l'école, Mme Isabel Amigo et M. Mathieu Simonnet qui furent nos encadrants de gestion de projet afin de nous accompagner et nous guider dans les différentes tâches dans la gestion du projet.
De l'autre côté, nos encadrants techniques, étant Mme Maï Nguyen, M. Christophe Lohr, et M. Maxime Devanne, font partie d'un groupe de recherche appartenant au laboratoire IHSEV (Interactions Humains Systèmes et environnements Virtuels) et M. Jacky Ménard qui appartient au département electronique. IHSEV travaille sur les interactions entre les utilisateurs et les systèmes avec deux grands domaines d'applications : l'assistance à la personne (Experiment'Haal) et les systèmes de mission attachés aux projets comme Keraal, Vitaal, Precious, Amalys, parmi d'autres.

Ce laboratoire est équipé d'un robot appelé Poppy Torso qui n'a pas la capacité de soulever des objets et donc ne peut pas les transporter. Notre objectif est au fournir au robot des mains préhensiles lui donnant la possibilité de pouvoir déplacer des petits objets afin que le laboratoire puisse utiliser cette nouvelle compétence pour démarrer de nouveaux projets.

Remerciements

Pendant le déroulement de ce projet nous avons reçu de l'aide de la part de nos différents encadrants, de gestion et techniques.
Ils nous ont prodigué des nombreux conseils et avis qui nous ont permis de mener à bien le projet. Tous ces conseils nous ont offert la possibilité d'avoir une vision en tant qu'ingénieur et nous ont fait réfléchir et poser les questions nécessaires.
Nous voudrions aussi remercier les responsables du Laboratoire Robotique de l'IMT Atlantique pour nous permettre d'y travailler. L'ambiance de ce laboratoire a toujours été favorable au développement nos différentes tâches.
Finalement, nous remercions notre partenaire extérieur qui nous a confié sa confiance dans la réussite de ce projet.

1. Cahier des charges

1.1 Contexte et enjeux du projet

Issu de la recherche française (laboratoire Flowers de l'INRIA Bordeaux), le robot Poppy Torso est l'une des principales créatures artificielles de la plate-forme technologique Poppy.

Poppy Torso, tout comme Poppy Humanoid, est un robot open-source (hardware et software), utilisant des pièces en impression 3D et des servomoteurs Dynamixel.

Le robot Poppy est avant tout un robot anthropomorphe : son design vise à reproduire la morphologie et les proportions d'un corps humain adulte. Cela implique un certain nombre de compromis : par exemple il n'est pas prévu pour marcher véritablement, et sa main n'est pas articulée.
Cependant la plate-forme Poppy s'adresse volontiers aux développeurs et aux contributeurs qui souhaitent améliorer ou étendre les fonctionnalités de Poppy.

En ce moment, le laboratoire IHSEV de IMT Atlantique possède un Poppy Torso, avec des mains qui ne sont pas articulées et pas préhensiles. La figure 1 montre ce robot avec ses mains anciennes.
Il y a en plus une Reactable (table interactive) avec un projecteur et une caméra infrarouge qui permet la lecture des codes Amoeba (QR codes) qui sont inclus dans le logiciel reacTIVision. La figure 2 représente la Reactable avec laquelle le travail a été fait.

Figure 1 : Robot Poppy Torso

Figure 2 : Reactable

1.2 Formulation du besoin

Comme il a été dit préalablement, le robot sur lequel le travail se pose n'a pas de mains pouvant prendre et transporter des objets. Ainsi, le principal objectif de notre projet est de désigner une nouvelle main pour Poppy Torso et la munir d'un moyen pour pouvoir saisir et bouger des petits objets.

Pour donner à ce but un contexte ou un environnement, un jeu simple a été développé pour que le robot puisse interagir avec un humain et pour qu'il puisse montrer sa nouvelle capacité de manipuler des objets légers. On se propose de le faire sur une Reactable. C'est une table qui a un projecteur et une caméra infrarouge intégrés qui lui permettent d'identifier et suivre des pions que l'on pose sur sa surface (dotés de codes appelés Amoeba codes). De plus, elle peut identifier des doigts et des blobs sans besoin d'un code particulier.

Les principales contraintes que ces objectifs présentent et auxquels nous avons dû faire face sont liées à des contraintes physiques. Parmi elles, ils peuvent être remarqués le poids maximum que le bras peut soulever sans forcer les moteurs ou encore la portée du robot sur la table.

1.3 Livrables et planification du projet

1.3.1 Livrables

Les principaux livrables de ce projet comprennent :

- Nouvelles mains pour Poppy, c'est-à-dire le design 3D des nouvelles mains et sa construction. Elles doivent être équipées avec des petits électroaimants.
- Commandes électroniques des électroaimants.
- Logiciel de Poppy pour le contrôle des électroaimants.
- Implémentation d'un jeu de plateau simple sur table Reactable entre Poppy Torso et un joueur humain (par exemple : tic tac toe).
- Une vidéo de démonstration.
- Mode d'emploi du système et son fonctionnement.

Avec l'accord des encadrants, les livrables intermédiaires suivants ont été proposés et ils permettent de tester chacune des parties séparément et de montrer l'avancement dans le projet :

- Design 3D de la main.
- Circuit de contrôle des électroaimants
- Nouvelle main pour Poppy avec contrôle sur les électroaimants.
- Démonstration de fonctionnement de la main avec les électroaimants
- Démonstration de robot qui déplace son bras d'un point A à un point B sur la table.

1.3.2 Planification du projet

Tout d'abord, avant commencer les tâches techniques il a fallu réfléchir sur des méthodologies et procédures à suivre.

Pour bien mener nos premiers pas dans la gestion des projets, six réunions avec des encadrants ont eu lieu pour nous guider vers un management efficace du temps et des ressources.

Pendant ces créneaux, un organigramme de tâches (WBS) qui résume toutes les activités à réaliser a été créé. Ce diagramme se présente dans l'annexe 8.1

Ensuite, chacune de ces tâches a reçu une durée estimée, avec lesquelles un diagramme de Gantt qui aidé à suivre les avancements et les délais de notre projet a été fait. La version finale de ce diagramme est montrée dans l'annexe 8.2.

Finalement, pour anticiper les possibles risques que le projet aurait pu subir, nous avons dessiné un tableau des risques sur lequel nous les spécifions et définissons de possibles solutions ou des actions de prévention afin d'atténuer leurs impacts.

2. Études et développement

2.1 Travail sur la main

L'objectif de ce lot est de faire une main préhensile capable de déplacer des objets légers. Dans le sujet du projet, il est proposé de la réaliser à l'aide d'un électroaimant. Donc l'idée initiale a été d'utiliser un électroaimant, mais en même temps, d'autres méthodes ont été conçus afin d'étudier différentes solutions. Les prochaines sections résument le travail fait. En plus d'avoir travaillé sur la main, des différents designs pour les pions ont été tenu en compte.

2.1.1 Modèle avec un électroaimant

La main originale du Poppy Torso n'est pas préhensile, c'est un morceau de plastique imprimé en 3D en forme d'une main humaine. Dans un premier temps, pour que le robot puisse saisir des objets légers, une main rectangulaire en carton avec un électroaimant au milieu a été faite afin de réaliser des tests. Considérant l'angle entre le bras du Poppy Torso et un plan horizontal, nous avons fait un connecteur simple qui permet à la main de se balancer d'un certain angle pour faciliter la prise avec le bras. Le modèle réalisé se trouve dans les figures 3 et 4 suivantes.

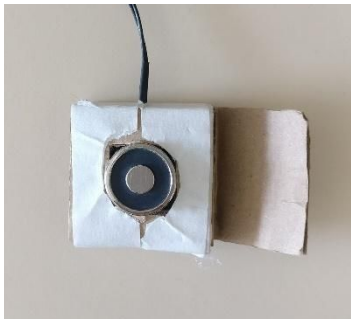


Figure 3 : Modèle simple de la main

Figure 4 : Modèle attaché au bras de Poppy

Après avoir réalisé ce modèle simple, nous avons pu tester l'électroaimant et voir comment la main fonctionne avec le bras. L'électroaimant utilisé [16] fonctionne avec une alimentation de 12V, le force est de 25N, autrement dit, il est capable de saisir un objet de 2.5kg. La première expérience assure que l'électroaimant est une méthode possible pour notre projet.
De plus, avec ce modèle, nous avons pu constater que le fait que la main soit plate, c'est-à-dire, que l'angle d'inclinaison entre le plan de la main et le bras soit fixé, n'empêche pas la main de prendre des objets sur toute la surface de la table.

2.1.2 Modèle avec des doigts actifs

Dans le premier modèle, l'électroaimant a bien fonctionné et Poppy pouvait saisir des objets légers. Après des discussions avec nos encadrants techniques,d'autres solutions ont été étudiées - une main préhensile avec des doigts actifs.

Des solutions proposées par d'autres sur Internet ont été prise en compte et après beaucoup de réflexions, et au vu du temps de projet limité, il a été choisi d'utiliser l'imprimante 3D de l'école pour faciliter la fabrication du modèle.

La première étape pour construire un modèle en utilisant une imprimante 3D est de faire un design en 3D, nous avons utilisé un logiciel gratuit - TinkerCAD, qui est offert par Autodesk. Il est facile d'apprendre à l'utiliser et il contient une riche bibliothèque de modèles. Il est suffisant pour construire notre modèle. Le modèle 3D réalisé est présenté dans la figure 5.

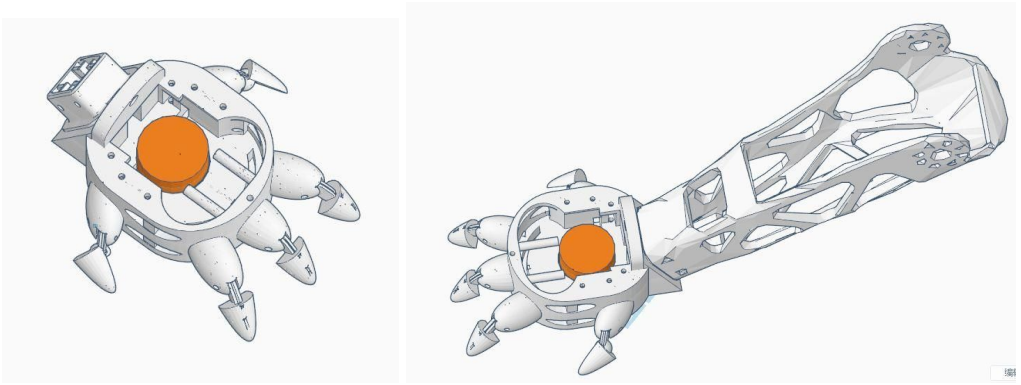


Figure 5 : Modèle 3D de la main avec les doigts actifs

Dans le modèle, il y a 5 doigts actifs, cela permet de faire ressembler la main à une main humaine. Le cylindre au milieu représente un électroaimant.

Une imprimante 3D a été utilisée pour en faire en PLA (plastique biodégradable) et en ABS (Acrylonitrile Butadiene Styrene plastic). Chacuns de ces deux matériaux a ses avantages et inconvénients[7].

Le PLA (Polylactic acid ou acide polylactique), biodégradable et issu de matériaux recyclés, peut être obtenu à partir d'amidon de maïs, par exemple. Il est notamment utilisé dans l'emballage alimentaire. Il y a moins d'erreurs lors de l'impression 3D. Le PLA sera plus adapté pour l'impression de pièces creuses ou avec de fines parois. Il ne nécessite pas de chauffer la plateforme d'impression, ce qui diminue le temps d'impression.

Mais le PLA est peu résistant à la chaleur, il commence à fondre à partir de 160℃, et est sensible à l'humidité. Il ne plie pas ou peu, ce qui le rend cassant. Plus difficile à travailler après impression, il ne pourra pas être limé ou percé par exemple.

L'ABS est un polymère thermoplastique, qui est notamment utilisé dans les appareils électroménagers. L'usage de ce matériau s'est illustré au travers de la fameuse brique de la marque Lego. L'ABS est plus résistant à la chaleur que le PLA, il commence à fondre à 180℃. Il se plie aussi plus facilement et ne rompt pas. Très résistant, il est adapté pour l'impression de pièces mécaniques par exemple.

Le seul bémol de l'ABS est que les risques d'échecs sont plus grands lors de l'impression, notamment pour des raisons de moindre résistance aux chocs de température. L'impression est également plus longue car la plateforme d'impression doit être maintenue aux alentours de 120℃ pour éviter les chocs de température.

Ces deux matériaux ont été testés et les modèles obtenus sont présentés dans la figure 6.

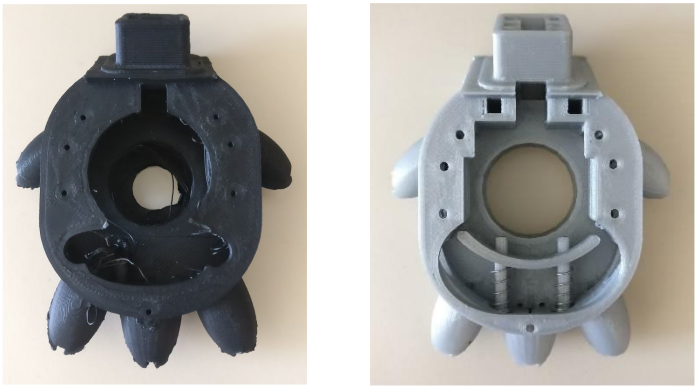


Figure 6 : Modèle en ABS(noir) et en PLA(gris)

D’après notre expérience, l’ABS ne marche pas bien pour notre main, il y a des problèmes comme des fissures, mais pour le PLA il n'y a aucun problème, donc celle-ci a été le matériel finalement utilisé.

Ayant déjà une main avec des doigts actifs en PLA, il y avait autre problème: les doigts ne peuvent pas se déplacer tout seul, ils ont besoin d'une source de mouvement. Différents dispositifs d'alimentation tels que des élastiques, des ressorts, des engrenages ou encore des fils fins ont été testés.

Les élastiques ne fonctionnent pas très bien, car la taille de la main et des doigts est petite, il est difficile de trouver de bons élastiques. Le même problème touche les ressorts, donc en fin de compte, nous avons utilisé des fils fins (fil de pêche) parce qu’ils ont la meilleure stabilité et ils sont plus simples à manier. Pour pouvoir contrôler les fils, un servo moteur a été mis sur le bras de Poppy et connecté les fils avec ce servo, une fois qu’il tourne dans le sens horaire, les fils se tendent, puis ils plient les doigts pour attraper l'objet. En tournant dans le sens antihoraire, les fils se détendent qui puis provoquent l'ouverture des doigts et lâchent l'objet.

Donc une version finale pour notre main préhensile est de fabriquer une main avec des doigts actif en PLA, avec les fils fins comme le dispositif d'alimentation. Une image de cette main est présentée sur la figure 7.

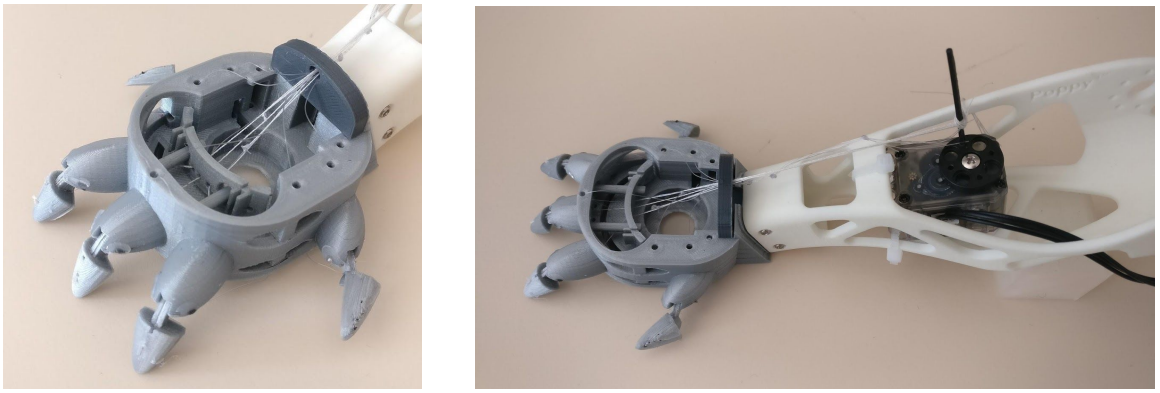


Figure 7 : Modèle final de la main de Poppy

En pratique, le dispositif mécanique ne fonctionne pas bien tout seul, il est possible de saisir un pion mais il est instable car les fils fins fournissent moins de force (<5N) et la surface de contact doigt-objet est faible (20mm²). De toute façon, même si nous ne sommes pas arrivés à faire fonctionner l'articulation des doigts, ces doigts donnent à la main un aspect plus anthropomorphe, ce qui est consistent avec la prémisse de la plateforme Poppy.

2.1.3 Design des pions

L'un des buts principaux de notre projet, est de faire jouer le robot à un jeu simple comme le tic-tac-toe. Pour pouvoir jouer, premièrement il a fallu construire les pions de jeu.

Envisageant le design final, il y avait quelques conditions que les pions devaient respecter :

- La forme devait être simple et plate pour que le robot puisse les saisir même sans articuler la main.
- Ils devaient contenir des pièces métalliques pour qu'ils puissent s'attacher à l'électroaimant.
- Le matériel devait être léger parce que l'électroaimant peut soulever jusqu'à un certain poids.
- Sur la base, il fallait coller des codes Amoeba alors la surface devait être suffisamment large pour contenir ces codes.

Après avoir étudié ces conditions, une première version des pions a été faite. Ces pions nous ont servi à tester l'électroaimant et à arriver à quelques conclusions sur des améliorations pour les prochaines versions. Cette fois-ci, les pions étaient carrés de 4cm de largeur en carton auxquels nous avons attaché des rondelles métalliques avec du ruban adhésif en papier d'un côté et un code Amoeba de l'autre. Cette version est montrée dans la figure 8.

Figure 8 : Première version des pions

Ces pions nous ont servi à vérifier le fonctionnement de l'électroaimant et aussi à nous familiariser avec les fonctionnalités de la Reactable. Nous avons pu commencer à repérer leur position et à étudier les réglages à faire sur les logiciels qui gèrent la Reactable.

Après quelques tests et analyses, la taille des codes n'était pas assez grande pour que la table puisse bien les identifier, alors des nouveaux pions ont été conçus, cette fois-ci ronds avec 6,5cm de diamètre et 0,5cm de hauteur. Pour le matériel la première idée était d'utiliser du plastique pour respecter le format des pions originaux mais à cause de l'indisponibilité de ressources, finalement le matériel utilisé a été le bois.
Dix pions ont été fabriqués, cinq pour le joueur humain et cinq pour le robot. Après leur avoir tous collé un code qui a été imprimé en papier, nous avons collé six rondelles métalliques au-dessus de ceux du robot avec de la colle thermofusible. La version finale est représentée sur la figure 9.

Figure 9 : Version finale des pions

2.2 Travail sur le robot

Au moment de commencer à travailler avec le robot, nous avons rencontré le premier problème. La carte Odroid XU4 du Poppy Torso était cassée. Alors rapidement il a été décidé d'acheter une carte Raspberry Pi 3 car c'était l'option moins chère et aussi parce qu'au laboratoire il y a d'autres projets qui utilisent cette carte déjà.

Ayant reçu la Raspberry Pi, quelques configurations ont été menés, suivant la documentation du Poppy Project :

- Télécharger l'image de *Raspbian Strech Lite* depuis le site officiel de Raspberry Pi (https://www.raspberrypi.org/downloads/raspbian/)
- Enregistrer l'image en utilisant le logiciel Etcher (https://etcher.io/) dans une carte SD
- Nous connecter par SSH à la Raspberry Pi avec un ordinateur qui utilise UbuntuMate 18.04 LTS.
- Élargir la partition de la carte SD:
  - `sudo raspi-config`
- Installer Raspoppy:
  - `curl -L https://raw.githubusercontent.com/poppy-project/raspoppy/master/raspoppyfication.sh | bash -s "poppy-torso"`
- Après l'installation et redémarrage, nous nous sommes connectés par SSH à la Raspberry Pi avec la nouvelle configuration:
  - `ssh poppy@<Poppy_IP_address>; login = poppy; Password = poppy`

Une fois la mise en marche de la carte Raspberry Pi terminée, nous devions modifier l'image pour pouvoir utiliser le module GrovePi+ et le Grove Electromagnet. Les instructions données par Dexter Industries ont été suivis:

- Cloner le dépôt avec les fichiers d'installation nécessaires pour le module GrovePi:
  - `cd /home/pi/`
  - `sudo git clone https://github.com/DexterInd/GrovePi`
  - `cd /home/pi/Desktop/GrovePi/Script`
- Exécuter le fichier d'installation exécutable:
  - `sudo chmod +x install.sh`
  - `sudo ./install.sh`
- Après le redémarrage, installer la librairie smbus pour être capables de contrôler l'électroaimant
  - `sudo pip install smbus`

À ce moment, nous avions une image capable de piloter le robot et aussi l'électroaimant. La seule chose à faire maintenant est de se connecter par SSH et créer un script Python pour les contrôler. Pour aboutir ce tâche, la librairie *pypot.robot* et un fichier .json avec la configuration de tous les moteurs (nom, numéro, référence, limites du mouvement) ont été

utilisés. Comme il a été dit préalablement, la librairie *grovepi* est celle utilisée pour piloter l'électroaimant.

Il faut souligner qu'il a fallu créer un nouveau fichier de configuration .json de manière d'être capables de gérer deux moteurs que ne sont pas dans la configuration original de Poppy Torso. Ces deux moteurs sont abs_x et abs_y. Le fichier de configuration peut être trouvé sur GitHub

2.3 Travail sur la Reactable

2.3.1 Prise en main de la Reactable

La Reactable (figure 8) est une table interactive qui se présente sous la forme d'une surface ronde ayant en-dessous une caméra infrarouge et un projecteur. La fonction principale de la Reactable est de créer des sons et de la musique de manière interactive avec des pions dotés de codes Amoeba (comme des QR codes) que la caméra infrarouge peut repérer.

Figure 10: Un modèle de Reactable

Au niveau logiciel, elle est séparable en deux éléments (voir figure 11 et figure 12). Une partie tracking, ici reacTIVision, qui va détecter les objets et leurs positions (et angles pour les Amoebas) ainsi qu'une partie client qui va écouter les informations ainsi envoyées et va afficher sur la table l'interface en fonction des messages.
Les messages sont envoyés à l'aide du protocole TUIO, souvent utilisé pour les interfaces tangibles. Il possède une implémentation sous Python que nous avons choisie pour faciliter la compatibilité avec l'application qui gère l'affichage graphique.

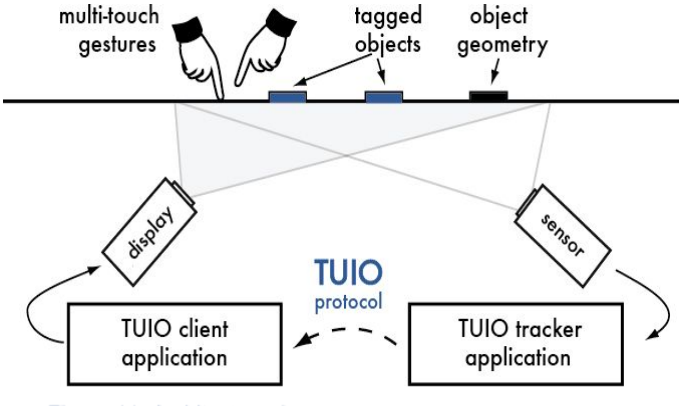


Figure 11: Architecture du protocole TUIO

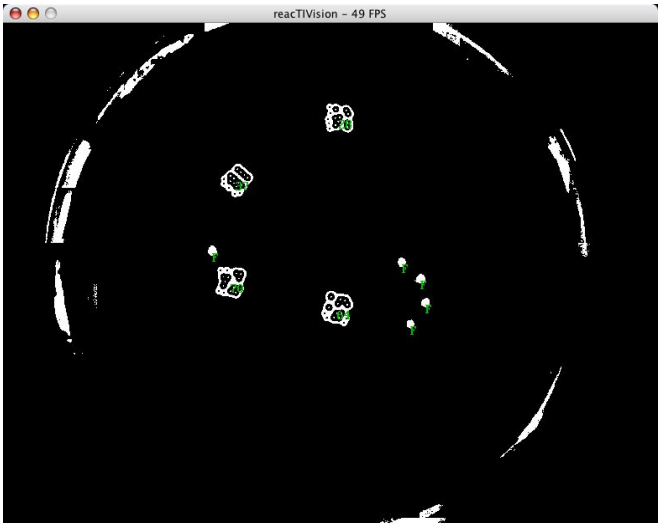


Figure 12: Interface reacTIVision avec une table interactive

2.3.2. Conception de l'interface graphique

L'interface graphique consiste ici en un plateau de jeu de tic tac toe simple afin que toute personne puisse comprendre où jouer sur la table.
Le plateau de jeu se décompose ainsi en une grille de 9 cases réparties sur 3 lignes et 3 colonnes dans lesquelles il faudra jouer les pions.
Cependant il y a plusieurs paramètres à prendre en compte. Le premier est qu'il faut prendre en considération la portée de Poppy qui peut difficilement atteindre le côté opposé à lui de la table (la table fait 85 cm de diamètre mais Poppy ne peut pas aller plus loin que 47 cm devant lui). Un autre est la précision de détection et la taille des pions qui vont fixer la taille minimale des cases. Effectivement, il ne faut pas que les cases soient plus petites que les pions car la prise de décision sur la case où se trouve le pion est alors compliquée et dépend bien plus de la précision de détection de la table.

Toute cette interface graphique est gérée par le script `board.py` à l'aide du module *Kivy* qui en plus de permettre de développer une interface dispose d'une implémentation de client TUIO ce qui permet d'écouter les messages envoyés lors d'événements sur la Reactable.

Le script `board.py` une fois éxécuté va lancer l'application qui va s'ouvrir dans une fenêtre sur l'écran qui sera projeté par en dessous de la Reactable. Le code va construire dans cette fenêtre notre première version simple du plateau de jeu.
Il se compose de 9 instances de la classe Button de Kivy disposés dans une grille de 3 lignes et 3 colonnes. La classe Button de Kivy est un widget "attaché" à la fenêtre de l'application sous la forme ici d'un rectangle dont la taille peut être changée, la couleur et auquel on peut lier une action lors d'un événement ayant lieu sur les coordonnées de ce bouton (par exemple les évenements *on_press*, *on_release*).
Dans notre cas lorsque qu'un pion sera posé sur l'un des ces boutons, il changera de couleur (bleu ou rouge selon le joueur) et signalera les coordonnées (ligne,colonne) du bouton pressé afin de pouvoir gérer le jeu et choisir le prochain mouvement du robot.

```
### We add the buttons on the board
### button dictionnary with name and state {button:{"id":int,"state":boolean}}

self.buttons={}
# Here We add the two relaunch buttons
self.add_widget(Button(background_color=(128,0,0,1),pos_hint={"center_x":0.400,"center_y":0.8},size_hint=(0.07,0.1)))
self.buttons[self.children[0]]={}
self.buttons[self.children[0]]["id"] = -1
self.buttons[self.children[0]]["state"] = False
self.add_widget(Image(source="board_text/j1.png",pos_hint={"center_x":0.400,"center_y":0.8},size_hint=(0.05,0.1)))

self.add_widget(Button(background_color=(0,0,128,1),pos_hint={"center_x":0.470,"center_y":0.8},size_hint=(0.07,0.1)))
self.buttons[self.children[0]]={}
self.buttons[self.children[0]]["id"] = -2
self.buttons[self.children[0]]["state"] = False
self.add_widget(Image(source="board_text/j2.png",pos_hint={"center_x":0.470,"center_y":0.8},size_hint=(0.05,0.1)))

# Here we add the 9 button of the board
buttonsNumber = 9
for i in range(buttonsNumber):
    row=(i)//self.cols
    col=(i)%self.cols
    x,y= (0.3+float(col)/8),(0.60-float(row)/7)

    self.add_widget(Button(background_color=(1,1,1,1),pos_hint={"center_x":x,"center_y":y},size_hint=(0.11,0.13)))
    self.do_layout()

    #self.children[0].bind(on_press=play)
    self.buttons[self.children[0]] = {}
    self.buttons[self.children[0]]["id"] = i
    self.buttons[self.children[0]]["state"] = False

print (self.buttons)
```

Figure 13 : Code de la création du plateau de jeu

Deux boutons, J1 et J2, ont été ajoutés permettant de relancer la partie à tout moment en choisissant d'être le premier ou le deuxième joueur. Ce sont les boutons que nous ajoutons à au début du code de la figure 13 avec la fonction `add_widget`.

La figure 14 montre la fonction codée `play` qui est liée aux neufs boutons du plateau de jeu. Elle va être appelée à chaque fois qu'un pion est détecté sur la position d'un des neufs boutons de jeu. Elle prend en argument l'instance Button du bouton pressé et retrouve alors la ligne et colonne du bouton pour l'écrire dans un fichier data qui sera lu continuellement par le script `gestion.py` afin de déterminer le prochain mouvement de Poppy.
Puis en fonction du tour de jeu elle détermine qui est le joueur qui vient de jouer et change la couleur du bouton.
Si le bouton est déjà pressé, la fonction le détecte et rien ne se passe.

```
### This sends the row and column of the button presssed if not already pressed
def play(button):

    ### We get the id, row and column of the button pressed
    i = self.buttons[button]["id"]
    row=(i)//self.cols
    col=2-(i)%self.cols

    ### We change color and return row,col if the button isn't already pressed
    if not(self.buttons[button]["state"]):
        if not(self.turn%2):
            button.background_color=(128,0,0,1)
        else:
            button.background_color=(0,0,128,1)
        self.turn+=1
        self.buttons[button]["state"]=True
        print('Turn: %s' %self.turn)
        print('The button %s (%s,%s) is being pressed' %(self.buttons[button]["id"],row,col))
        f = open('data', 'w')
        f.write(str(self.turn)+'\n')
        f.write(str(row)+'\n')
        f.write(str(col)+'\n')
        f.close()
        ssh = paramiko.SSHClient()
        ssh.load_host_keys(os.path.expanduser(os.path.join("~", ".ssh", "known_hosts")))
        ssh.connect("10.77.3.120", username="poppy", password="poppy")
        sftp = ssh.open_sftp()
        sftp.put("data", "/home/pi/poppy_project/game_code/data")
        sftp.close()
        ssh.close()
        pass
    else :
        #print('Turn: %s' %self.turn)
        #print('The button %s (%s,%s) is already pressed' %(self.buttons[button]["id"],row,col))
        pass
```

Figure 14 : Code de la fonction play

Cependant même avec un code correct nous avons remarqué dès la première utilisation de la Reactable qu'il y avait un décalage dans les coordonnées envoyées par la Reactable et celle de l'application gérée par Kivy. Particulièrement sur l'axe des x cartésiens nous avons remarqué un décalage vers la gauche. C'est à dire qu'un pion posé sur la Reactable sera positionné avec un décalage d'une certaine longueur vers la gauche par l'application.
Après un certain temps de recherche sur l'origine et sur une correction potentielle de ce phénomène nous avons conclue que qu'il était lié à la position du projecteur.

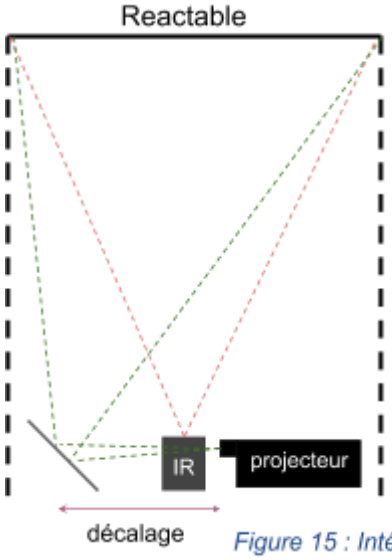


Figure 15 : Intérieur de la Reactable

En effet le projecteur est posé horizontalement sous la Reactable et fait face à un miroir qui renvoie l'image sur la surface de la Reactable alors que la caméra infrarouge est posée verticalement (figure 15). Le décalage vient du chemin à parcourir entre le projecteur et le miroir. Le décalage sur l'axe des y peut être corrigé avec la fonction trapèze du projecteur mais celui sur l'axe des x était trop grand. Pour le corriger physiquement nous l'avons pris en compte dans le code de `board.py`. Ce décalage n'est pas constant alors il a été corrigé linéairement dans le code. A chaque événement sur la table le programme récupère les coordonnées de l'instance et les corrige puis cherche "manuellement" si elle se situe sur un bouton pour savoir quelle fonction appeler.

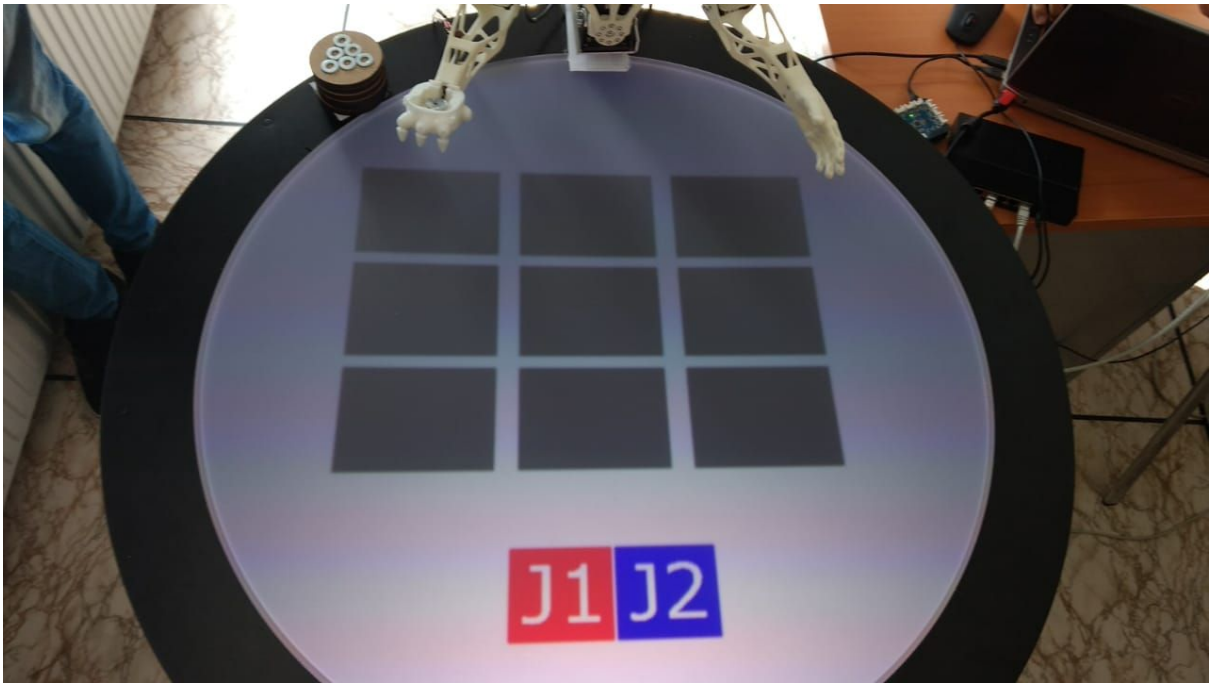


Figure 16: Le plateau de jeu affiché sur la Reactable

2.3.3 Quelques chiffres et précisions

Pour pouvoir désigner le plateau du jeu, il a fallu tout d'abord prendre des mesures de la table et de la portée du robot. La figure 17 montre un schéma de la table où il est possible de voir les dimensions de cette portée, celles du plateau et les positions initiales du robot et ses pions.

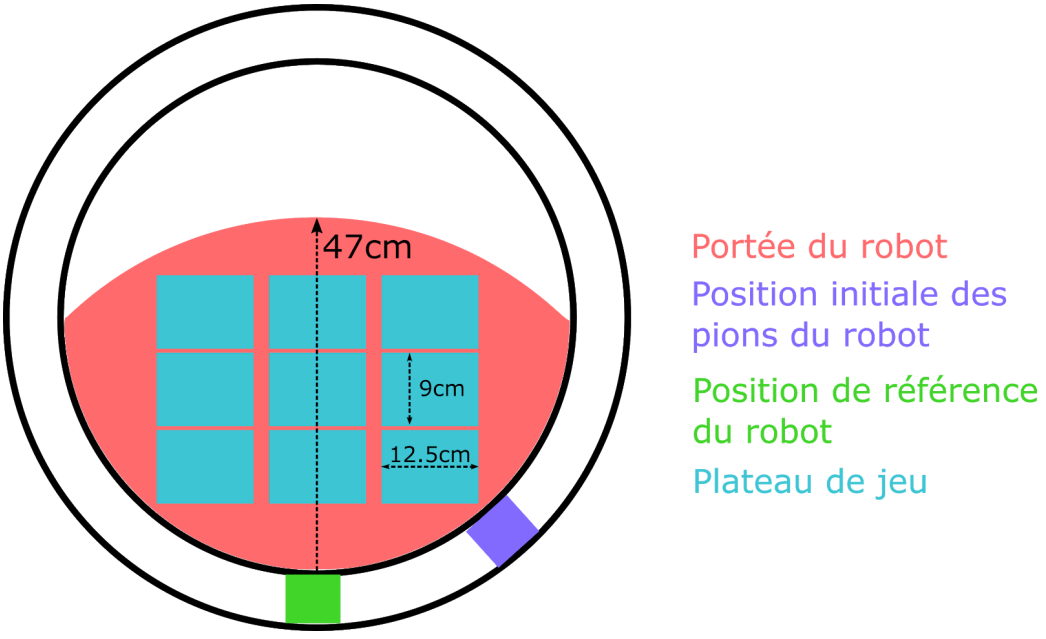


Figure 17 : Situation de la table

De plus, après les réglages faits pour corriger les décalages des coordonnées, la précision obtenue continue à dépendre de la coordonnée «*X».*

Pour mesurer la précision avec laquelle la table détecte les pions nous avons comparé la position du centre des pions avec la position détectée qui est aussi affichée.
Cet écart est mineur pour les coordonnées x de faible valeurs. Le décalage va de 1,5cm à 4,5cm.

2.4 Développement du jeu

Comme il a été dit précédemment, le jeu développé est le tic tac toe. Cette décision a été prise car le type de mouvements que le robot peut faire est compatible avec le type de mouvements que le jeu demande. En plus, son implémentation n'est pas complexe.

Sachant qu'il existe beaucoup de versions du jeu choisi, ici se trouvent les règles sur lesquels le développement a été fait.
Tout d'abord, le but du jeu est d'arriver à mettre trois pions propres sur une même ligne, colonne ou diagonale d'un plateau qui consiste en une grille de 3x3. Les joueurs font ses mouvements par tours et ainsi le premier joueur a l'avantage de jouer avec un pion de plus (cinq en lieu de quatre).
Le premier joueur qui arrive à aligner trois de ses pions gagne la partie.

Le code du jeu a été développé en python affichant le plateau sur l'écran de nos ordinateurs pour pouvoir analyser le code et réaliser des simulations.
Au début, pour faciliter les tâches, le code a été programmé de façon que le robot joue aléatoirement, bien-sûr vérifiant si la case choisie est disponible.
Puis, des améliorations ont été pilotés en donnant au robot la capacité de jouer d'une manière plus *« intelligente »*. Dans cette version, le robot peut identifier s'il y a des conditions gagnantes tant pour lui que pour l'opposant. Ainsi, il peut bloquer les cases de l'opposant dans le cas où il serait sur le point de gagner et inversement se rendre compte quand il sera sur le point de gagner. De cette façon, le jeu est maintenant plus réel et dynamique.

Ces codes sont sur le GitHub [11]. Afin de mieux les comprendre, ils sont expliqués ici.
Le code principal du jeu se trouve dans le script python `gestion.py`. Ce code s'exécute localement dans la RaspberryPi et il contient tous les éléments pour que le jeu puisse fonctionner. En définitive:

- Le code importe la librairie pypot.robot et puis, avec elle, nous créons un objet poppy qui vais contrôler tous les moteurs du robot à partir d'un fichier de configuration .json modifié. De plus, nous définissons que le robot ne sera pas en mode compliant. C'est-à-dire, avec tous les moteurs allumés.
- Le code importe la librairie grovepi pour gérer l'électroaimant avec la carte GrovePi+
- Le code importe les fonctions de deux autres fichiers créés:

    - `strategy.py` : contient toutes les fonctions utilisées pour la gestion du jeu.
    - `movements.py` : contient toutes les fonctions et déclarations pour les mouvements du robot et de l'électroaimant.
- Le code déclare les variables globales du jeu:
    - Une matrice 3x3 que contient l'information qui arrive de la Reactable pour prendre décisions et faire le jouer le robot.
    - Strings identifiants, les différents états que chaque case de jeu peut avoir et sont représentés dans la matrice (`empty, player_1, player_2`).
    - Un compteur pour l'historique des coups joués.
    - Deux variables, `current_player` et `current_winner`, qui sauvegardent le tour de joueur actuel et s'il y a un vainqueur ou non.
    - Variables de type boolean pour la gestion du jeu.

Après les déclarations, le script continue dans une boucle continue qui gère le tour de l'humain et du robot, pour obtenir les informations du plateau de jeu, le script utilise une boucle de polling qui interroge un fichier qui est transmis par SSH depuis l'ordinateur qui exécute le code de la table vers la Raspberry Pi, chaque fois qu'un pion est placé dans une boîte de jeu.

Ce fichier contient 3 lignes avec un entier dans chacune d'elles, la première indique le numéro de jeu, la seconde contient la rangée où le pion de jeu a été posé et la troisième, la colonne.

Lorsque le joueur humain réinitialise le jeu et choisit de jouer en premier ou en second, le fichier contient un -1 comme numéro de tour et dans la deuxième ligne le numéro de joueur du robot.

Le script quittera la boucle de polling dans deux conditions:

- Lorsque le joueur réinitialise le jeu, envoyant un -1 comme numéro de tour
- Quand une pièce est jouée par le joueur humain.

Avant de quitter la boucle, le script mettra à jour le numéro de tour avec le numéro renseigné par le Reactable, et après avoir agi, il ne quittera plus jusqu'à le joueur humain joue à nouveau. De plus, il place la valeur correspondante dans les variables booléennes pour effectuer les actions correspondantes.

- Si le joueur a réinitialisé le jeu, la fonction de `reset` sera exécutée:
    - Vide la matrice
    - Définir le numéro de joueur sélectionné pour le robot
    - Initialiser `n_play = 0`
    - Définir `current_winner` comme vide
    - Définir `current_player` comme `player_1`
    - Initialiser la position du robot
    - Retour à la boucle d'interrogation
- Si le joueur a joué un pion:

    - Mettre à jour la matrice avec la position du pion du joueur, dans la fonction `play`
    - Valider si le joueur a gagné (ou s'il y avait une égalité), avec la fonction `check_win`
    - Si le joueur a gagné (ou qu'il y a eu une égalité) : revenir à la boucle de polling, en attendant que le joueur réinitialise le jeu, en sélectionnant un numéro de joueur.
    - Si le joueur n'a pas gagné, donne le tour au robot
- Quand le tour est donné au robot:
    - Faire le bon choix, le choix dépend des fonctions du code `strategy.py` expliquées ci-dessous.
    - Comme le robot effectue initialement des mouvements aléatoires, il doit être vérifié que la case qu'il souhaite occuper est vide avec la fonction `validate_move`
    - Mettre à jour la matrice
    - Valider si le robot a gagné (ou s'il y avait une égalité).
    - Effectuer le mouvement du robot, avec les fonctions de `movements.py`
    - Revenir à la boucle de polling, en attendant l'action du joueur, si le robot a gagné il doit y avoir une réinitialisation et sinon le joueur peut continuer à jouer.

En ce qui concerne les mouvements du robot, dans le fichier movements.py il y a des fonctions très simples qui nous permettent de déplacer le robot en fonction du tour à réaliser

La fonction `robot_play` sélectionne le pion que le robot doit prendre en fonction du numéro de joueur et du numéro de tour, les mouvements pour prendre chaque pion sont différents puisqu'ils sont empilés d'un côté au début du jeu, donc la coordonnée z changera à chaque tour du robot.

Chaque fonction `take_piece` déplace le bras du robot à une position au-dessus de la pile de pions, puis l'abaisse pour le positionner exactement au-dessus du pion à mettre en jeu, et enfin active l'électroaimant.

Ensuite, la fonction `move_to_board` prend simplement la ligne et la colonne que le script `strategy.py` a sélectionné, déplace le bras du robot jusqu'à une position centrale, puis sélectionne le vecteur d'angles moteur nécessaire pour déposer le pion dans la case sélectionnée et utilise finalement ce vecteur pour déplacer le robot et éteindre l'aimant.

Les fonctions `move_robot` et `em_control` sont extrêmement basiques, la première utilise simplement un attribut de l'objet `poppy`, créé avec le fichier `.json` et la bibliothèque `pypot.robot` pour déplacer le robot. La seconde utilise la bibliothèque `grovepi` pour effectuer une écriture numérique (allumer ou éteindre) le port 4 du GrovePi, où l'aimant Grove Electromagnet est connecté.

Par rapport à la stratégie du robot, dans le fichier `strategy.py` Il y a quatre types des fonctions qui sont décrites ici.

- Fonctions qui vérifient s'il y a des conditions gagnantes:

Ces fonctions parcourent les lignes, les colonnes et les diagonales et elles comptent la quantité de symboles correspondant au même joueur et donnent comme résultat un drapeau qui sera vrai quand il y a une ligne, colonne ou diagonale où il y a deux symboles du même joueur. En plus, elles renvoient les variables `aux_O` et `aux_X` qui contiennent le numéro de la ligne ou la colonne où il y a la condition gagnante à exception des fonctions qui font leur travail sur les diagonales pour lesquelles ce n'est pas nécessaire car il y a une seule diagonale principale et une seule diagonale secondaire. C'est-à-dire que ces fonctions peuvent identifier si l'un des joueurs est sur le point de gagner, et ils détectent la ligne, colonne ou diagonale sur laquelle il faut agir au prochain tour.

- Fonctions qui trouvent la case à compléter:

Quand les fonctions précédentes déterminent qu'il y a une ligne, colonne ou diagonale qui peut faire gagner quelqu'un, ces fonctions trouvent la case où il n'y a pas de pion, c'est-à-dire, elles informent la prochaine position à jouer.

- Fonctions qui déterminent le mouvement à faire:

À partir des informations obtenues des fonctions précédentes, ces fonctions donnent par résultat les variables `row` et `col` qui définissent le mouvement à faire. En tenant compte de la situation où les deux joueurs peuvent gagner et si c'est le tour du robot, le mouvement qu'il doit faire c'est celui qui l'amène à la victoire. Pour savoir si le robot est représenté par les croix ou les cercles le code regarde s'il joue en premier ou en second.

- Fonction qui fait le mouvement:

Finalement, la fonction `make_move` vérifie s'il y a quelque drapeau levé et si c'est le cas elle définit le mouvement rendant les variables `row` et `col` avec le mouvement que le robot doit faire. Si ce n'est pas le cas, le robot jouera de manière aléatoire.

2.5 Intégration

2.5.1 Intégration du code du jeu avec la Reactable

Initialement, pour tester le bon fonctionnement du code du jeu, le plateau est affiché sur la fenêtre d'exécution du code et les coordonnées correspondantes aux tours du joueur humain sont introduites manuellement.

Cependant, l'objectif est que l'information sur la situation du plateau soit apportée par la Reactable en temps réel. Alors, pour pouvoir lier l'information provenant de la table avec le code du jeu, un fichier qui contient les coordonnées et le numéro du dernier tour est créé. Ce fichier est mis à jour chaque fois que la table détecte un nouveau pion et il est envoyé par SSH à l'ordinateur qui mène la gestion du jeu.

2.5.2 Intégration du code du robot avec le code du jeu et la Reactable

Avec l'information du numéro de tour et en sachant si le robot joue en premier ou en deuxième, on obtient les tours du joueur humain et ainsi met à jour le plateau virtuel et décide sur le mouvement que le robot va faire.

Comme il a été dit dans la section précédente, le fichier `board.py` utilise la librairie Kivy qui est chargée de lire l'information donnée par la caméra sous la Reactable. Puis, il envoie un fichier qui contient les données de chaque pièce jouée. Cette information est utilisée pour la gestion du jeu.
Avec l'information concernant l'état du plateau du jeu, une ligne et une colonne sont générées par le fichier `strategy.py`.
Ensuite, les angles correspondant à chaque moteur, pour chaque position finale que le bras doit avoir, ont été enregistrées dans différents vecteurs.
Finalement, avec la ligne et la colonne, la matrice du jeu est mise à jour, la gestion du jeu est réalisée et les vecteurs du mouvement de chaque moteur sont choisis.

3. Matériaux mis en jeu

Pour le développement de notre projet les matériaux utilisés, à part de ceux que le laboratoire nous a fourni (le robot Poppy et la Reactable), sont ceux utilisés pour la fabrication de la main et des pions.

Pour concevoir et fabriquer la nouvelle main et les pions, pas seulement des ressources matérielles ont été utilisés mais aussi des logiciels et des outils appartenant au Laboratoire de Fabrication de l'école (FabLab).

Ici, tous ces matériaux sont listés.

- Logiciel et outils

Pour le design de la main, c'est-à-dire pour construire le modèle 3D qui a été imprimé par la suite, le logiciel TinkerCAD a été utilisé.
Puis, pour adapter notre design à l'imprimante 3D, nous avons dû utiliser le logiciel Cura.

L'outil du FabLab utilisé pour imprimer la main est l'imprimante 3D dont le modèle est Ultimaker 2+/3.
Pour découper les pions, l'outil utilisé est un découpeur Laser.

- Matériaux

Les matériaux utilisés lors de la fabrication de la main sont :

  - Filament gris PLA 1.75mm (100-200g)
  - Filament noir ABS 2.85mm (100-200g)
  - Fil de pèche (<1m)
  - Ecrous type N1 (6)
  - Ressorts (3)
  - Bois (490 cm²)

- Matériaux achetés
  - 1 paquet de rondelle métalliques modèle n°7: 1.5 €
  - 1 carte microSD 16GB: 18.6 € [12]
  - 2 électroaimants de type Grove Pi+: 20.8 € [13]
  - 1 carte Grove Pi+: 30.79 € [14]
  - 1 carte Raspberry Pi 3 modèle B 1G: 36.95 € [15]
  - 2 électroaimants de 12V, 25N pour faire des tests: 4.34 € [16]

- Temps employé dans le FabLab
  - Temps pour imprimer une main: 5h
  - Temps pour préparer le modèle des pions pour le découpeur laser: 1h
  - Temps pour découper les pions: 10m

4. Validation et tests

Les commandes électroniques de l'électroaimant ont été bien conçues et réalisées. Ils se trouvent dans le code `movements.py`. Plus d'informations sur le contrôle de l'électroaimant se trouve dans la section 2.4.
Concernant le logiciel de Poppy, l'image a été modifiée de manière à être capable de piloter le module GrovePi+, ce module est en charge de commander l'électroaimant.

Deux types d'électroaimant différents ont été testés. Le premier électroaimant fonctionne sous une alimentation de 12V, il offre une maximale force de 25N, donc il est capable de saisir les choses moins de 2.5kg théoriquement.
Il fonctionne bien avec les pions réalisés, mais il a besoin d'une alimentation différente que celle de Poppy.
L'autre électroaimant est un module de Grove-Electromagnet qui fonctionne avec une alimentation de 5V, avec une force fournit de 10N, même si cet électroaimant a moins de force, cela suffit pour notre projet. En plus, il est capable de fonctionner sur la carte de Grove Pi, donc c'est plus facile de le contrôler en utilisant la Grove Pi.

Chacune des mains a été testé avec les pions finaux afin de choisir la meilleure.

Un vidéo de démonstration d'une partie jouée avec le robot ainsi qu'un mode d'emploi jouent le rôle de livrable de validation.

5. Résultats et perspectives

Pour chacun des différents lots de tâches menés, nous avons eu de différents résultats.

Pour le lot de la main, nous avons réussi à obtenir une solution qui s'adapte aux besoins, qui est capable de saisir des objets comme il l'aurait été demandé et même si encore actuellement elle n'est pas visuellement semblable à une main humaines, on envisage de la faire ressembler autant que possible en ajoutant des doigts dont on pourra se servir pour donner au robot la capacité de les articuler et ainsi, prendre des objets non métalliques.

Par rapport au robot, nous avons bien pu maîtriser ses moteurs et c'est prouvé par le fait de pouvoir donner au robot les ordres nécessaires pour le faire bouger de la façon désirée. C'est évident que selon notre méthodologie de travail, il faut connaître au début les position finales de chaque mouvement.

En ce qui concerne la table, après les difficultés que rencontrées, l'image faisant le rôle de plateau et consistant en neuf boutons est bien affichée. Finalement, la synchronisation de la position physique des objets sur la table avec ce que la caméra lit sur la position de ces objets a bien été faite.
C'est à dire que les décalages rencontrés dans le sens des *«x»* et des *«y»* cartésiens ont été réduits en tant que possible.

L'ensemble des livrables (vidéo, mode d'emploi, liste de matériels, modèles 3D...) se trouvent sur notre repository GitHub ainsi que la totalité des codes utilisés.

https://github.com/ftdargentina/2018_s4_projet13

6. Conclusion

À l'aboutissement du projet le jeu se déroule correctement avec Poppy et il peut réaliser ses mouvements de manières autonome et saisir des pions aimantés. La figure 18 montre le robot au-dessus de la table jouant.

Tout le travail préalable comprenant une étude des besoins du client, l'établissement de notre plan de travail et d'une solution qui nous semblait pertinente, nous ont permis d'arriver à ce résultat optimal et d'accomplir nos objectifs dans les temps donnés.

Nous sommes conscients que nous avons eu une opportunité unique de connaître le métier d'un ingénieur sachant que ce sera le type de travail que nous devrons affronter dans le futur. De plus, le fait d'être dirigés et orientés par des spécialistes est une bonne façon d'en faire une première approche.

Ce projet n'est que la prémisse à de futurs projets touchant au domaine de l'interaction entre un humain et un robot. Même si ce n'était pas inclus dans les objectifs principaux du projet, nous avons bien essayé d'incorporer un moteur à la main pour l'articuler et de cette façon pouvoir manipuler des objets qui ne soient pas nécessairement métalliques. Cependant par manque de temps, nous avons abandonné cette idée la laissant comme incitation à continuer dans le futur avec un autre projet.

Le prototype réalisé ce semestre pourrait très bien être la base d'un futur projet visant à faire réaliser à Poppy des tâches à la place d'un humain.

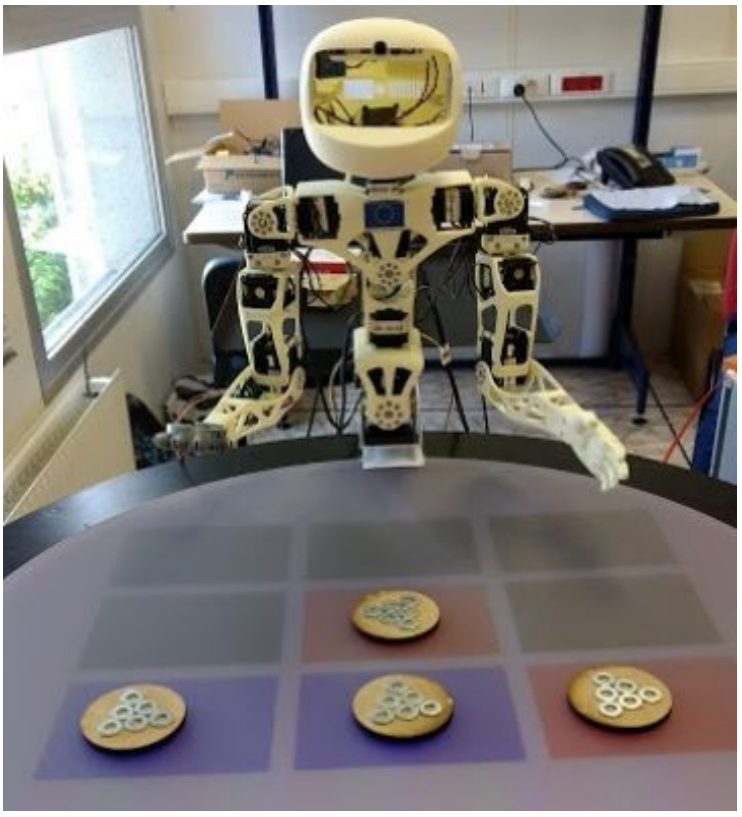


Figure 18 : Poppy jouant au tic tac toe

7. Références

[1] Menant D., Yvinec S. *Rapport de projet MGP 320*: *Interface tangible domotique*. IMT Atlantique, campus de Brest, 2017, 19p.

[2] Menant D., Yvinec S. *Dossier de conception*: *Interface tangible domotique*. IMT Atlantique, campus de Brest, 2017, 13p.

[3] Menant D., Yvinec S. *Documentation du code: Interface tangible domotique*. IMT Atlantique, campus de Brest, 2017, 9p.

[4] Grizou J., Lapeyre M., Rabault N., et al. Poppy Project, (2015), GitHub repository, <*https://github.com/poppy-project*>

[5] IMT Atlantique. RoboLabo [en ligne]. Disponible sur: <http://wiki-robot.enstb.org/doku.php?id=poppy>

[6] Dexter Industries. Setting up the software [en ligne]. Disponible sur <https://www.dexterindustries.com/GrovePi/get-started-with-the-grovepi/setting-software/>

[7] Autodesk. Formation 3d France [en ligne]. Disponible sur <https://www.formation-3d-france.com/comparatif-pla-et-abs/> (Consulté le 12/06/2018).

[8] Reactable. Reactable expérience [en ligne]. Disponible sur <http://reactable.com/experience/> (Consulté le 12/06/2018).

[9] Martin Kaltenbrunner. TUIO [En ligne]. Disponible sur <https://www.tuio.org/?members> (Consulté le 12/06/2018).

[10] ReacTIVision. ReacTIVision 1.5.1. Disponible sur <http://reactivision.sourceforge.net/> (Consulté le 12/06/2018).

[11] Dadam F. et al. 2018_s4_projet13, (2018), GitHub repository, <https://github.com/ftdargentina/2018_s4_projet13>

[12] Lextronic. [En ligne], consulté le 17 avril 2018, disponible sur <https://www.lextronic.fr/stockage-memoire/12233-carte-memoire-microsdhc-16gb.html?search_query=evo16gb&fast_search=fs>

[13] Lextronic. [En ligne], consulté le 17 avril 2018, Disponible sur <https://www.lextronic.fr/modules-divers/30480-module-électroaimant.html?search_query=101020073&fast_search=fs>

[14] Lextronic. [En ligne], consulté le 17 avril 2018, disponible sur <https://www.lextronic.fr/shield-fomat-grove/30485-platine-d-interface-grove-pi.html?search_query=103010002&fast_search=fs>

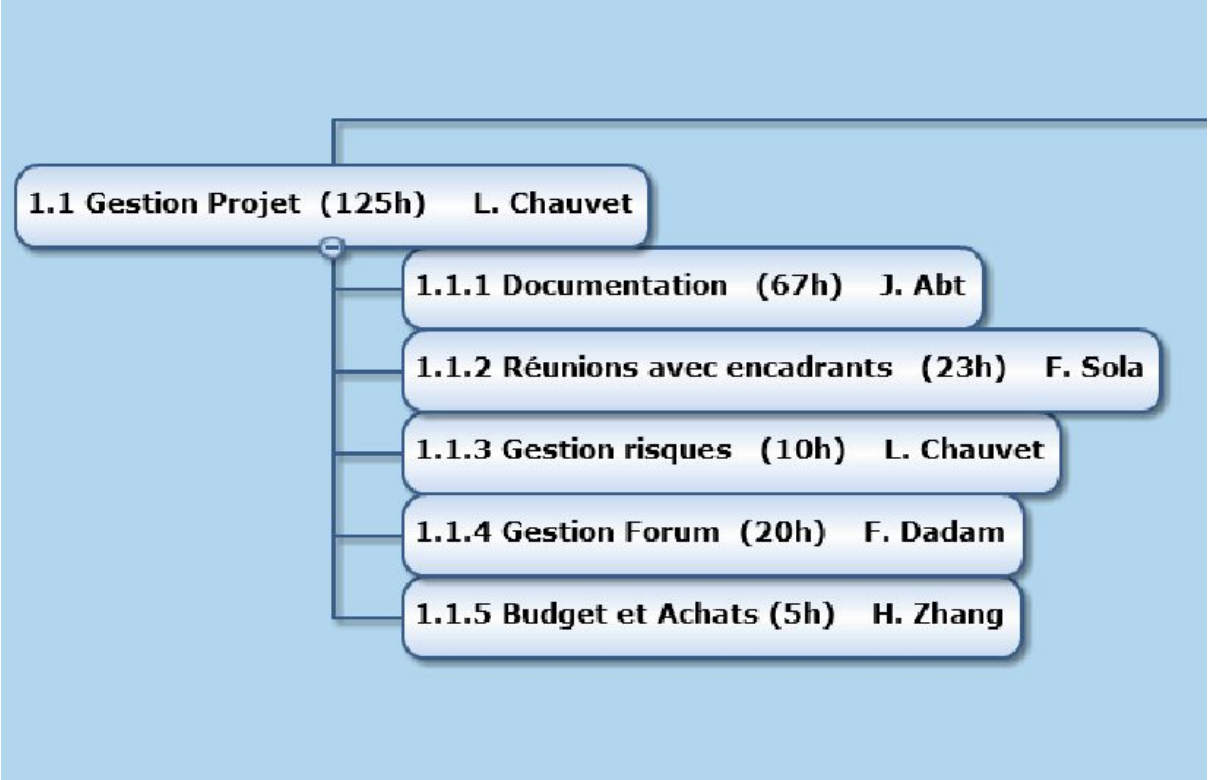
[15] KUBII. [En ligne], consulté le 17 avril 2018, disponible sur <https://www.kubii.es/raspberry-pi-3-2-b/1628-raspberry-pi-3-modelo-b-1-gb-kubii-64052710850.html>

[16] Amazon. [En ligne], consulté le 12 avril 2018, disponible sur <https://www.amazon.fr/électroaimant-SODIAL-leve-personne-electrique-maintien/dp/B01DXBCW8U>

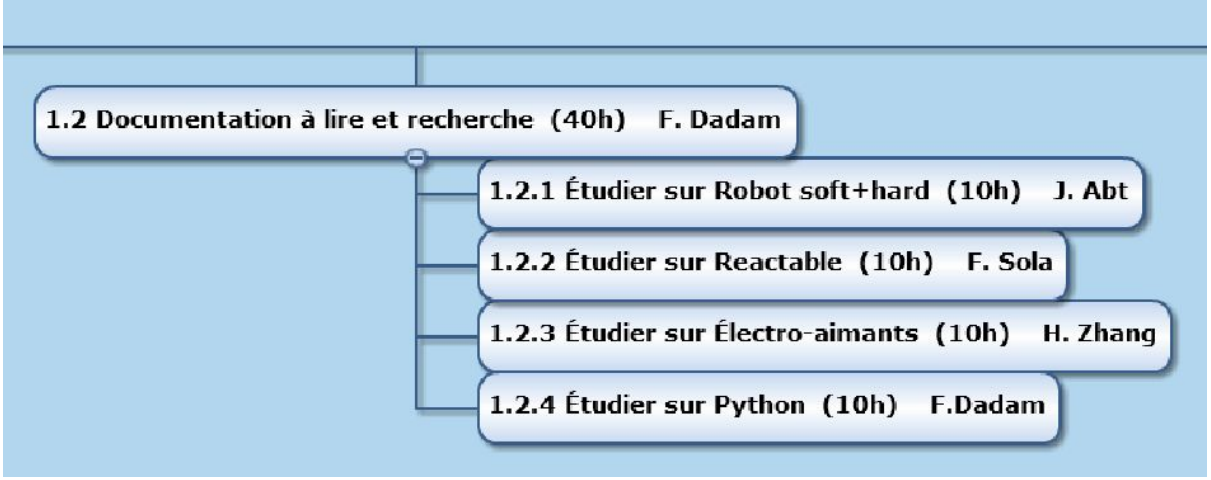
8. Annexes

8.1. Organigramme de tâches (WBS)

8.1.1. Gestion projet

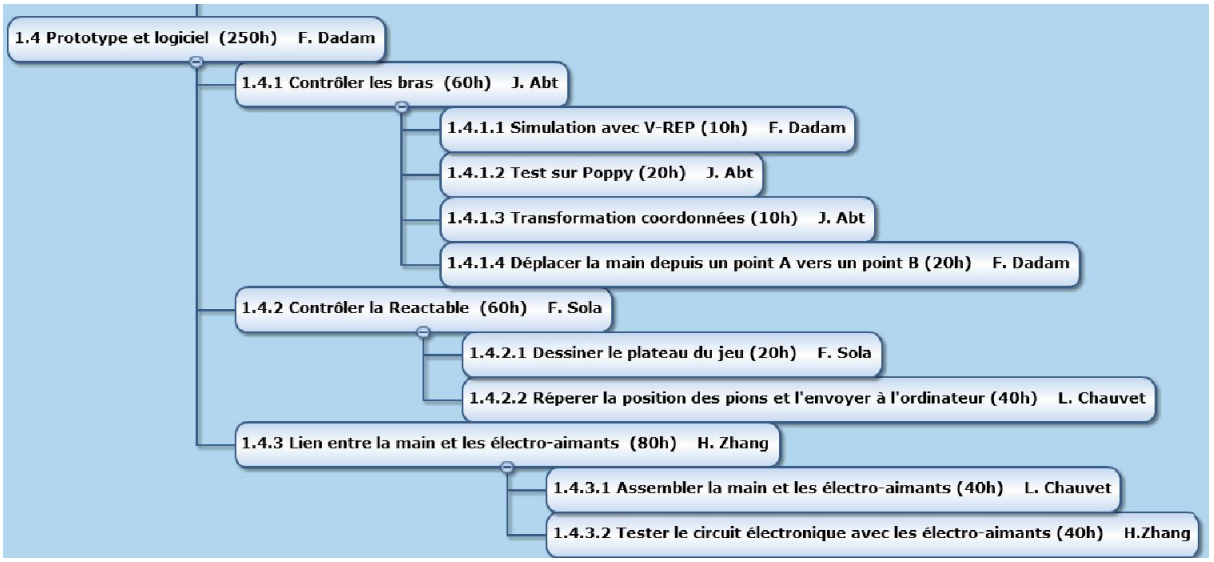


8.1.2. Documentation à lire et recherche



8.1.3. Analyse des besoins

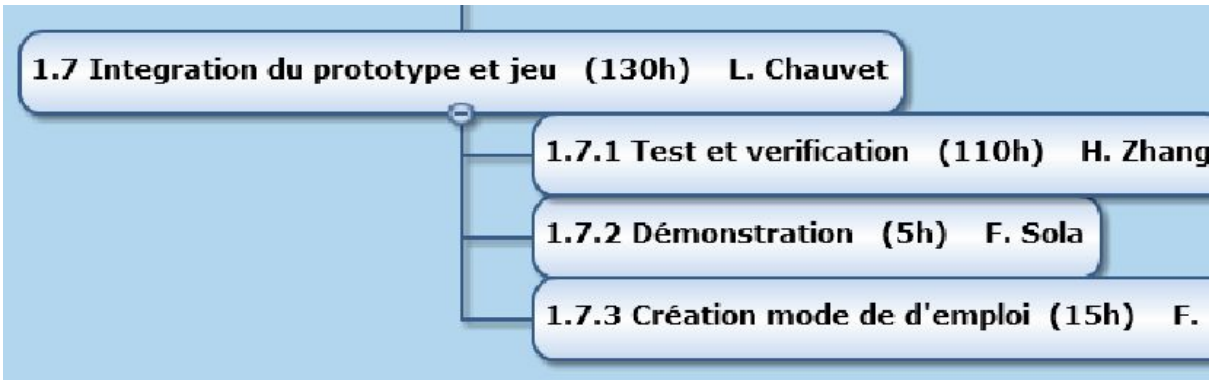
8.1.4. Prototype et logiciel



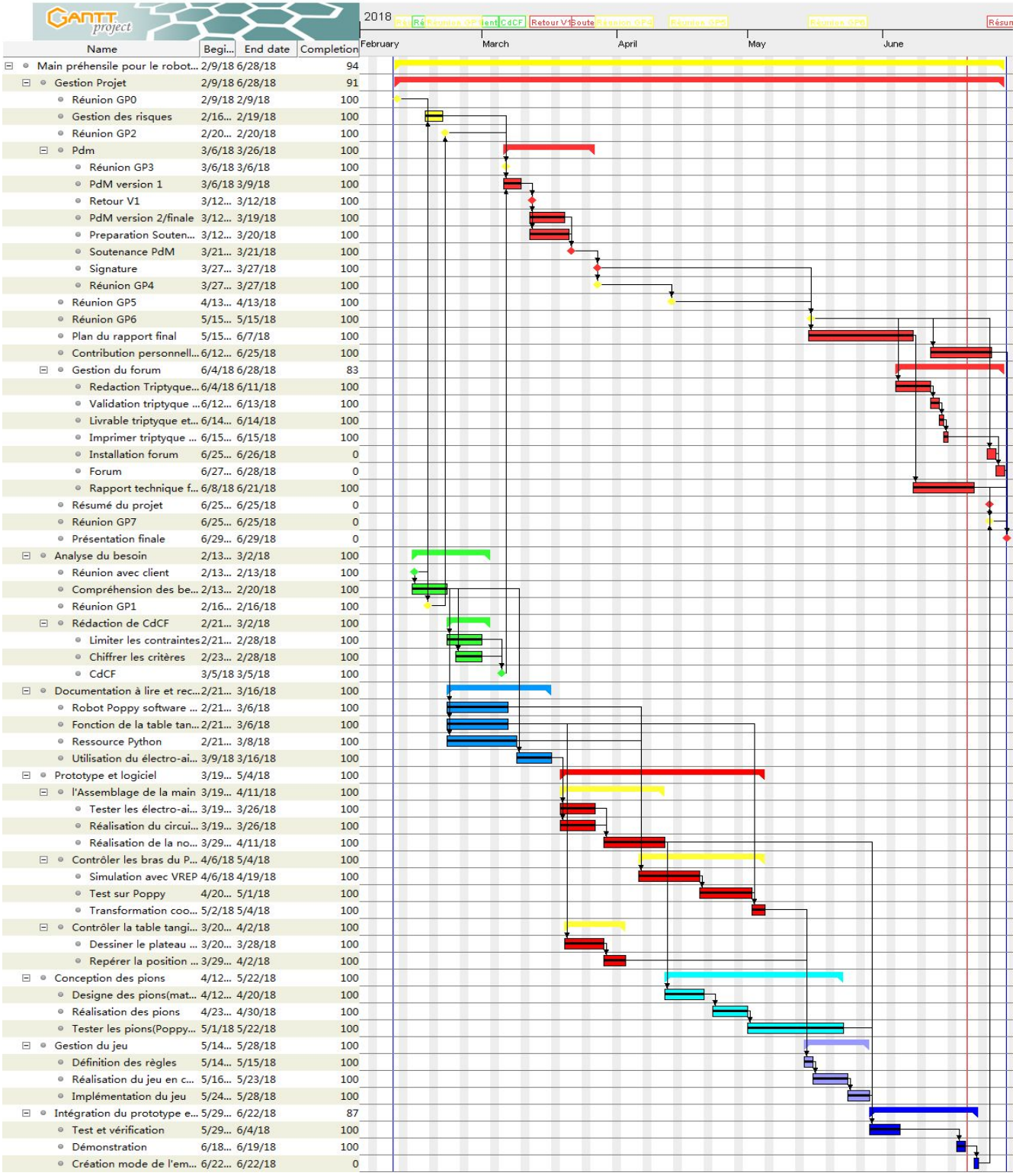
8.1.5. Design

8.1.6. Gestion du jeu

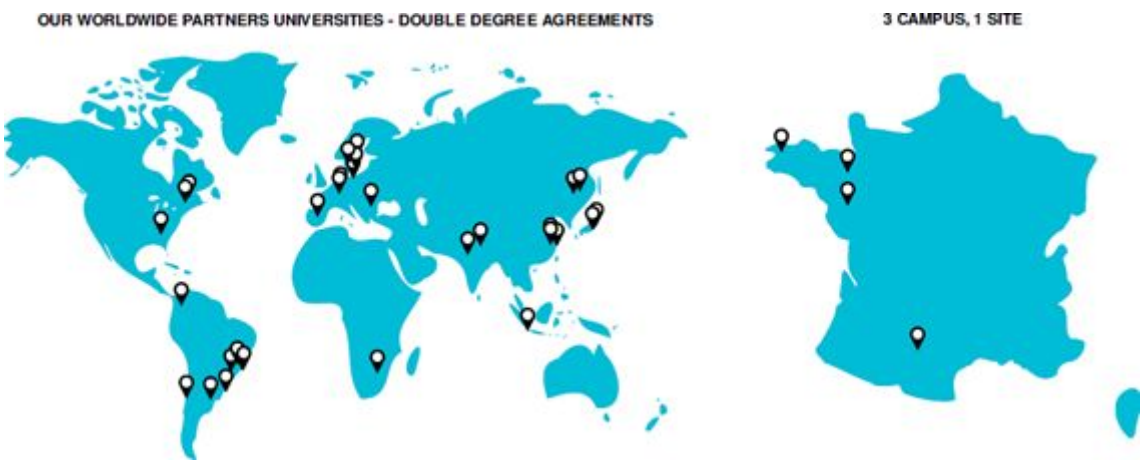
8.1.7. Intégration du prototype et logiciel



8.2. Diagramme de Gantt



Name	Begi...	End date	Completion
Main préhensile pour le robot...	2/9/18	6/28/18	94
Gestion Projet	2/9/18	6/28/18	91
Réunion GP0	2/9/18	2/9/18	100
Gestion des risques	2/16...	2/19/18	100
Réunion GP2	2/20...	2/20/18	100
Pdm	3/6/18	3/26/18	100
Réunion GP3	3/6/18	3/6/18	100
PdM version 1	3/6/18	3/9/18	100
Retour V1	3/12...	3/12/18	100
PdM version 2/finale	3/12...	3/19/18	100
Preparation Souten...	3/12...	3/20/18	100
Soutenance PdM	3/21...	3/21/18	100
Signature	3/27...	3/27/18	100
Réunion GP4	3/27...	3/27/18	100
Réunion GP5	4/13...	4/13/18	100
Réunion GP6	5/15...	5/15/18	100
Plan du rapport final	5/15...	6/7/18	100
Contribution personnell...	6/12...	6/25/18	100
Gestion du forum	6/4/18	6/28/18	83
Redaction Triptyque...	6/4/18	6/11/18	100
Validation triptyque ...	6/12...	6/13/18	100
Livrable triptyque et...	6/14...	6/14/18	100
Imprimer triptyque ...	6/15...	6/15/18	100
Installation forum	6/25...	6/26/18	0
Forum	6/27...	6/28/18	0
Rapport technique f...	6/8/18	6/21/18	100
Résumé du projet	6/25...	6/25/18	0
Réunion GP7	6/25...	6/25/18	0
Présentation finale	6/29...	6/29/18	0
Analyse du besoin	2/13...	3/2/18	100
Réunion avec client	2/13...	2/13/18	100
Compréhension des be...	2/13...	2/20/18	100
Réunion GP1	2/16...	2/16/18	100
Rédaction de CdCF	2/21...	3/2/18	100
Limiter les contraintes	2/21...	2/28/18	100
Chiffrer les critères	2/23...	2/28/18	100
CdCF	3/5/18	3/5/18	100
Documentation à lire et rec...	2/21...	3/16/18	100
Robot Poppy software ...	2/21...	3/6/18	100
Fonction de la table tan...	2/21...	3/6/18	100
Ressource Python	2/21...	3/8/18	100
Utilisation du électro-ai...	3/9/18	3/16/18	100
Prototype et logiciel	3/19...	5/4/18	100
l'Assemblage de la main	3/19...	4/11/18	100
Tester les électro-ai...	3/19...	3/26/18	100
Réalisation du circui...	3/19...	3/26/18	100
Réalisation de la no...	3/29...	4/11/18	100
Contrôler les bras du P...	4/6/18	5/4/18	100
Simulation avec VREP	4/6/18	4/19/18	100
Test sur Poppy	4/20...	5/1/18	100
Transformation coo...	5/2/18	5/4/18	100
Contrôler la table tangi...	3/20...	4/2/18	100
Dessiner le plateau ...	3/20...	3/28/18	100
Repérer la position ...	3/29...	4/2/18	100
Conception des pions	4/12...	5/22/18	100
Designe des pions(mat...	4/12...	4/20/18	100
Réalisation des pions	4/23...	4/30/18	100
Tester les pions(Poppy...	5/1/18	5/22/18	100
Gestion du jeu	5/14...	5/28/18	100
Définition des règles	5/14...	5/15/18	100
Réalisation du jeu en c...	5/16...	5/23/18	100
Implémentation du jeu	5/24...	5/28/18	100
Intégration du prototype e...	5/29...	6/22/18	87
Test et vérification	5/29...	6/4/18	100
Démonstration	6/18...	6/19/18	100
Création mode de l'em...	6/22...	6/22/18	0



IMT Atlantique Bretagne – Pays de la Loire – http://www.imt-atlantique.fr/

Campus de Brest
Technopôle Brest-Iroise
CS 83818
29238 Brest Cedex 3
France
T +33 (0)2 29 00 11 11
F +33 (0)2 29 00 10 00

Campus de Nantes
4, rue Alfred Kastler
CS 20722
44307 Nantes Cedex 3
France
T +33 (0)2 51 85 81 00
F +33 (0)2 99 12 70 08

Campus de Rennes
2, rue de la Châtaigneraie
CS 17607
35576 Cesson Sévigné Cedex
France
T +33 (0)2 99 12 70 00
F +33 (0)2 51 85 81 99

Site de Toulouse
10, avenue Édouard Belin
BP 44004
31028 Toulouse Cedex 04
France
T +33 (0)5 61 33 83 65

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom