

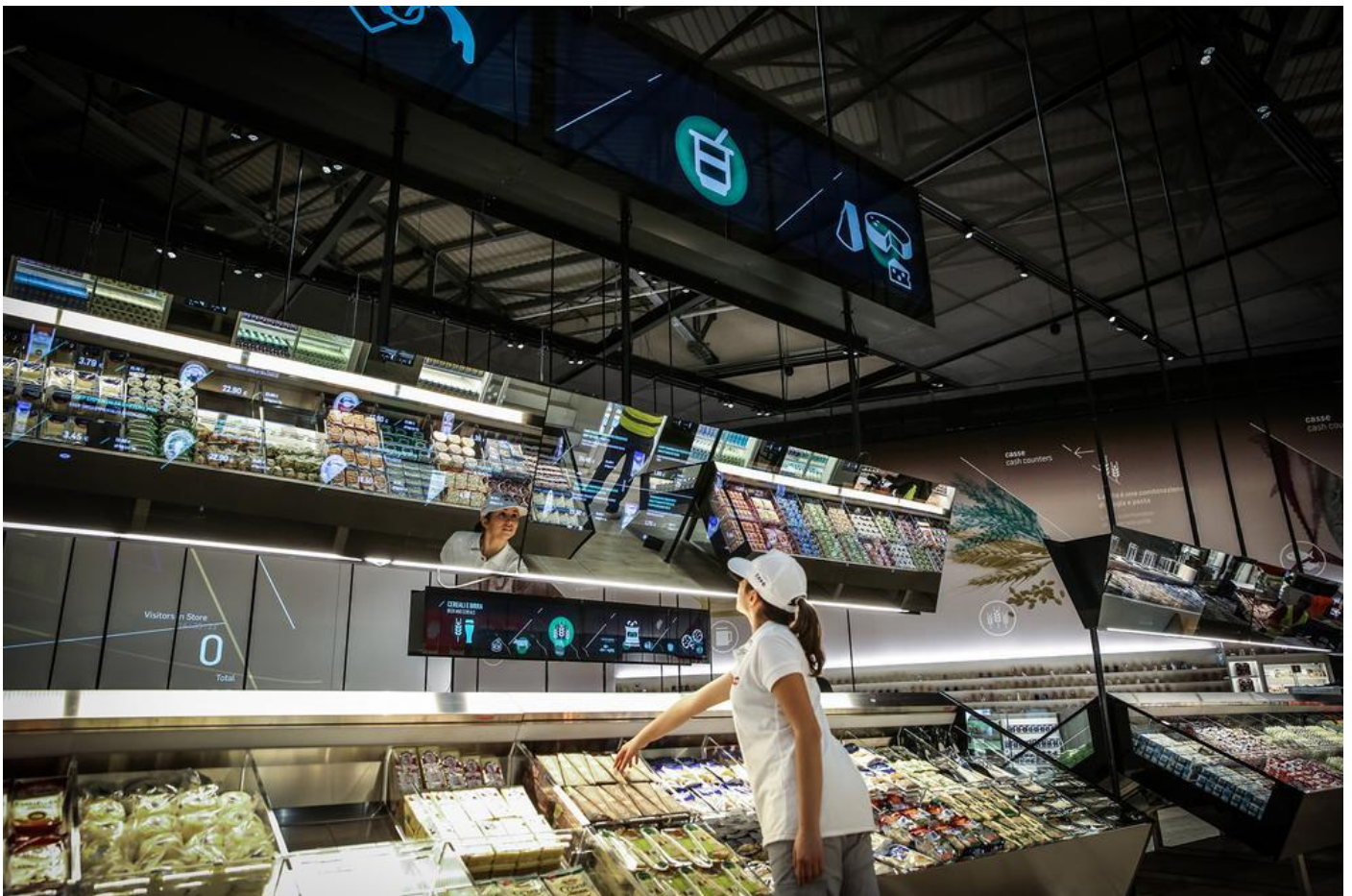
FOLL-E

RAPPORT FINAL

AVRIL 2017

PPE 1667

Référents : M. Jae Yun JUN KIM, M. Pierre PARADINAS



Groupe : Étienne BLANC, Adrien CHEVRIER, Wen CUI, Bastien HARENDARCZYK, Quentin MURET

REMERCIEMENTS

Tout d'abord nous adressons nos remerciements à notre mentor durant ce PPE, Monsieur Jae Yun JUN KIM, qui par sa disponibilité et son expertise a su répondre à l'ensemble de nos questions ainsi que nous aiguiller pour mener ce projet à son terme.

Nous tenons aussi à remercier vivement Monsieur Pierre Paradinas, le responsable de la valorisation de notre projet, grâce à son expérience nous avons su à qui nous adresser pour mieux comprendre les enjeux de notre projet et son secteur d'application vis-à-vis du grand public.

Nous remercions également Monsieur Julien Cottin, directeur d'enseigne chez Carrefour, pour avoir accepté de nous recevoir en entretien et donner de son temps pour juger de la valeur de notre projet pour une application dans une enseigne comme la sienne. Il nous a ainsi permis de penser notre projet dans un environnement plus concret.

Nous adressons également nos remerciements à M. Daniel Buruian pour nous avoir guidés dans le choix de notre matériel,

M. Nicolas Menigoz pour avoir échangé avec nous sur le projet,

M. Bouchez pour avoir supervisé le PPE,

Nos familles respectives pour nous avoir offert un soutien moral du début à la fin du projet,

Et enfin toutes les personnes qui ont pu contribuer de près ou de loin au succès de ce projet et à son aboutissement.

Sommaire

Remerciements	3
Introduction	5
Méthodologies et organisationN.....	6
1. Organisation.....	6
2. Application Android.....	7
3. Balance	9
4. Suivi d'utilisateur	12
I. Résultats.....	18
1. Résultats obtenus.....	18
2. Discussion sur les résultats	26
Conclusion.....	27
Bibliographie.....	28
Liste exhaustive des produits du projet accessibles.....	28
Table des Figures.....	29

INTRODUCTION

L'automatisation de l'encaissement au supermarché se popularise de plus en plus avec le l'émergence des caisses automatiques, et dans une moindre mesure avec les lecteurs de code barre intégrés aux chariots. Ces deux solutions obligent le client à attendre à la caisse et sortir ses articles du panier après avoir terminé ses achats.

Notre projet consiste à apporter une alternative permettant de payer en sortant du magasin sans avoir à décharger les articles. Étant intéressés par la robotique et la programmation mobile nous avons conjointement décidé de créer le chariot sous forme de robot piloté par une application mobile sous Android.

Le système consiste en une application smartphone utilisée pour scanner les articles et les payer, et un chariot composé d'une balance permettant de peser l'ensemble des articles scannés pour vérifier que les achats correspondent au contenu du chariot (permettant ainsi d'éviter les vols).

Ce chariot communique avec le smartphone de l'utilisateur. Ce dernier servira à scanner les produits et effectuer le paiement en fin de course.

L'utilisateur pourra aussi, via son application, consulter la liste des articles présents dans le caddie et obtenir des informations dessus, comme par exemple la liste des ingrédients ou encore le taux d'acides gras. Le paiement sur le caddie même permet ainsi de supprimer entièrement l'étape d'encaissement en sortie de magasin.

Le chariot est également capable de suivre l'utilisateur de manière autonome grâce à divers capteurs. Le client peut ainsi se déplacer dans le magasin, faire ses achats et payer librement sans se soucier des charges à porter, sans avoir à attendre ou à pousser son caddie.

Les fonctions primaires :

robot	smartphone
mesurer le poids du contenu avec fiabilité	Scanner le code barre des produits
	Récupérer et afficher les données relatives aux produits (prix, ingrédients...)
envoyer les données relatives au poids au smartphone	Vérifier que le poids pesé correspond au poids théorique des produits
	Faire payer l'utilisateur pour ses produits

Les fonctions secondaires :

robot	smartphone
suivre l'utilisateur	Stocker les informations de consommation du client
	Permettre au client de consulter les articles présents dans son caddie
	Proposer un itinéraire d'achats intelligent au client
	Proposer des produits en fonction des promotions actuelles et habitudes du clients

METHODOLOGIES ET ORGANISATION

1. Organisation

Nous avons séparé le projet en différentes parties de manière à se répartir les tâches. Ainsi chacun s'est vu attribuer différents objectifs en fonction de ses capacités et de sa spécialité :

Bastien s'est principalement occupé de monter la balance et de la faire communiquer avec le smartphone. Adrien s'est chargé mettre en place les moteurs, les asservir en fonction des différents capteurs et mettre en place la communication entre la Raspberry et le téléphone en plus de son rôle de chef de projet. Quentin a quant à lui développé l'algorithme côté application pour récupérer le poids de la balance et le comparer au poids théorique afin de vérifier que les produits achetés sont bien dans le caddie. Etienne s'est occupé de toute la partie paiement via l'application ainsi que de la gestion de la base de données servant à récupérer les informations sur les produits et la mise en place du scanner de code barre avec reconnaissance automatique. Wen a eu pour rôle de localiser le chariot par rapport à l'utilisateur grâce à différents capteurs Bluetooth.

Partie chariot	Partie application
Adrien	Quentin
Bastien	Etienne
Wen	

Figure 1: Répartition des tâches

Sprint	Missions
Sprint 1	- Cahier des charges
Sprint 2	- Cahier des charges - Choix des composants - Prise en main du matériel
Sprint 3	- Développement application - Mettre en pratique la balance
Sprint 4	- Développement application - Mettre en pratique la balance - Suivi d'utilisateur
Sprint 5	- Développement application - Montage chariot - Suivi d'utilisateur

Figure 2: L'avancement du projet

2. Application Android

Nous avons choisi de développer l'application sur le système mobile Android pour deux raisons principales : la popularité du système (Android est l'OS le plus répandu au monde) et pour des raisons d'accessibilité aux ressources de développement. En effet Android étant un système open-source il existe beaucoup de bibliothèques et d'API disponibles notamment pour le Bluetooth et le scan de code barre.

L'application est développée sous Android 6 pour être compatible à la fois avec une majorité de téléphone Android et pouvoir disposer des dernières technologies (surtout pour la partie Bluetooth afin de disposer de la technologie BLE – 4.1).

Bluetooth



La Raspberry et l'application se connectent en Bluetooth 4.1. Dès l'ouverture de l'application celle-ci cherche à se connecter à la Raspberry. Une fois la connexion établie le bouton scan se « débloque », permettant à l'utilisateur de scanner un article. À chaque scan l'article est en attente et le bouton de scan bloqué tant que la Raspberry n'a pas renvoyé une valeur de poids qui correspond à l'article inséré.

Une fois que l'utilisateur a fini ses courses, celui-ci doit procéder au paiement. Une fois le paiement terminé, l'application envoie à la Raspberry l'instruction d'allumer une LED présente sur le caddie. Cette dernière est un indicateur visuel permettant d'indiquer au vigils de sortie de magasins que le caddie a été payé ou non.

Un socket peut se traduire par « connecteur réseau » ou « interface de connexion ». C'est un élément logiciel qui est aujourd'hui répandu dans la plupart des systèmes d'exploitation. Il s'agit d'une interface logicielle grâce à laquelle un développeur peut facilement exploiter les services d'un protocole réseau.

Pour se connecter à la Raspberry pi, nous utilisons directement son adresse mac, présente dans le socket, ainsi qu'un UUID (pour Universally Unique IDentifiers). Cet identifiant doit être le même côté serveur (Raspberry) et côté client (Application Android).

Pour communiquer, nous utilisons une socket RFCOMM, ce qui permet une communication en streaming (pas d'accusé de réception). On parle aussi de communication en port série. Ainsi, pour envoyer ou recevoir des données, nous devons impérativement passer par ce port série. Nous avons donc établi un Protocole entre l'application et la Raspberry permettant le bon fonctionnement de l'application.

Les requêtes que peut envoyer le smartphone à la Raspberry sont les suivantes :

Requête	Action
quit	Fermeture et destruction du socket de communication.
led	Le paiement a été effectué, la balance doit contrôler que le client n'ajoute pas de produits dans le caddie. Si c'est le cas, une led est allumée, sinon une led clignote.
Autre	La RPI renvoie le poids de la balance.

Scan

Le scan est géré par une application tierce entièrement intégrée et transparente pour l'utilisateur dans notre application. Lorsque le bouton de scan est pressé, une instance de l'appareil photo s'ouvre. Si elle détecte un code barre dans son champ de vision, elle le décode automatiquement et le renvoie à notre appli, qui se charge d'instancier le produit et de lui attribuer ce code barre.

Récupération des données produit

À partir du code barre l'application envoie une requête HTTP à un serveur privé sur lequel se trouve la base de données des produits du magasin. Le serveur, appelé « middleware » permet de traiter les requêtes de l'application Android en requête SQL et renvoyer le résultat de chaque requête en un format interprétable par l'application (JSON). Le prix, le poids, la photo, les descriptions, le nom et tout autre donnée désirée sont alors assignés à ce produit.



Figure 3: Fonctionnement de l'échange de données entre l'application et le serveur

3. Balance

Pour des raisons de sécurité liée au vol de produits, nous avons développé une balance permettant de vérifier le poids du produit scanné.

Hardware

Pour réduire le cout de notre balance, nous nous sommes servis d'une jauge de déformation pour récupérer le poids du produit.

La jauge de déformation, est un capteur passif de type résistif. En fonction du poids placé sur la jauge, nous obtenons une variation de la tension. Pour illustrer cela, nous avons caractérisé le capteur. En appliquant une tension de 5V en entrée du capteur, et en faisant varier le poids appliqué sur ce dernier, nous avons obtenu le graphique suivant :

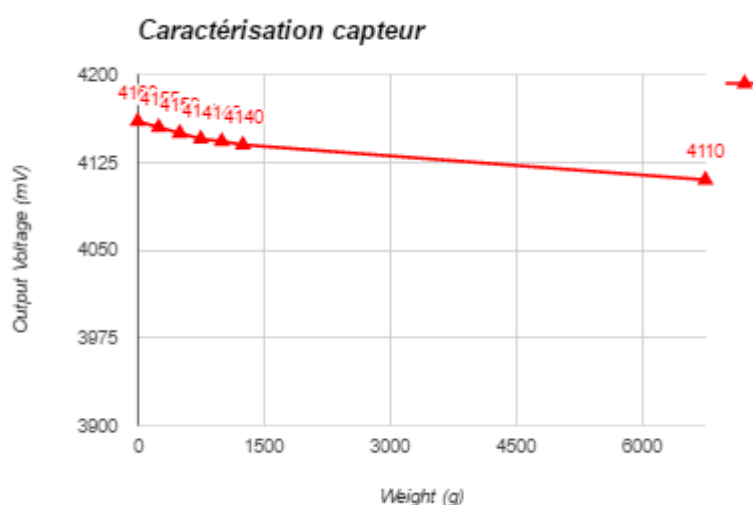


Figure 4: Caractérisation du capteur.

Étant donné la faible variation, il a fallu que nous mettions en place un système permettant de récupérer ces variations de manière précise.

Pour cela, nous avons utilisé la chaîne de traitement suivante :

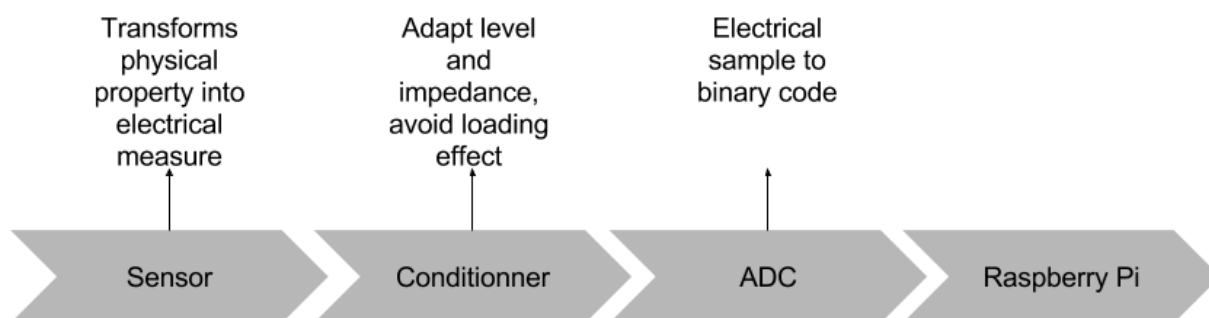


Figure 5: Synoptique de la Balance (BARBOT, 2015).

Sensor

La partie sensor comprend la jauge de contrainte et son montage de mesure. Dans notre cas le montage utilisé est un pont de Wheatstone. Le pont de Wheatstone est utilisé pour mesurer une résistance électrique inconnue par équilibrage de deux branches d'un circuit en pont, avec une branche contenant le composant inconnu. Dans notre cas, le composant inconnu correspond à la jauge de contrainte...

Le conditionner

Il met en forme le signal mesuré pour le traduire en une grandeur permettant le traitement. En effet, il adapte le niveau et l'impédance du système, tout en éliminant le loading effect. Dans notre système, le conditionner est composé d'un filtre passe-bas, permettant l'élimination du bruit, et d'un amplificateur.

Le filtre passe-bas est un circuit passif de premier ordre. Un circuit passif est un circuit composé de composants passifs (Résistances et condensateurs) qui ne nécessitent pas de source d'énergie externe.

Le convertisseur analogique-numérique

Le convertisseur nous permet de convertir notre grandeur physique (mesurand), qui est la tension, en une valeur numérique. Pour choisir le bon ADC, nous nous sommes intéressés à plusieurs caractéristiques : la précision et la plage de tension. Dans notre cas, la plage de tension en entrée est de 0-5V. D'après notre caractérisation du capteur, nous devons détecter une variation de l'ordre du mV. Nous avons donc choisi un ADC 12bits. Cet ADC nous permet de détecter une variation de 1.2mV :

$$\Delta U = (5-0) / 2^{12} = 1.2\text{mV}$$

Cette détection minimum nous permet de détecter un changement de poids au niveau du caddie.

L'ensemble de ces parties nous donne le système suivant :

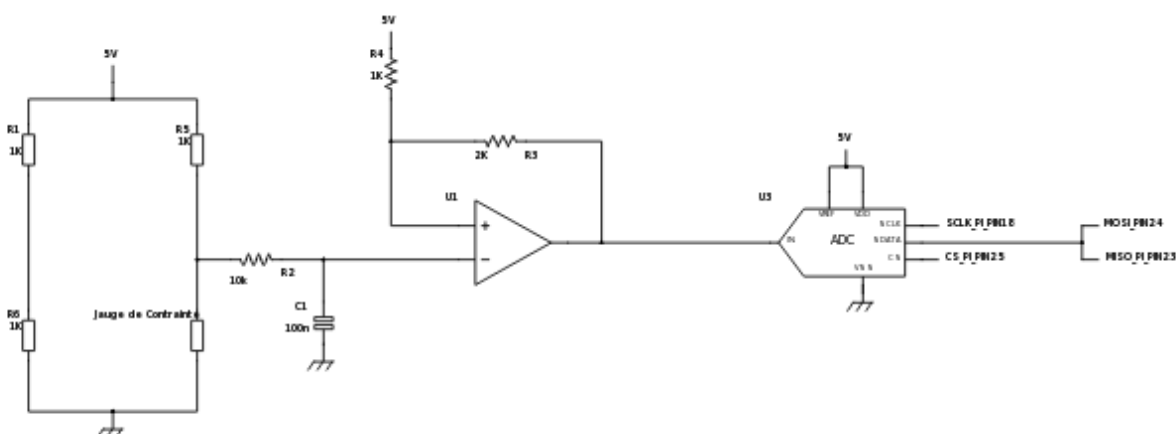


Figure 6: Système de la Balance.

Ainsi, nous avons notre schéma hardware de la balance.

Software

Une fois le schéma hardware de la balance réalisé, nous avons codé son application sur la Raspberry pi. La partie software doit permettre de récupérer la valeur renvoyée par l'ADC, de la traiter et de l'envoyer via Bluetooth au smartphone.

Communication SPI ADC-Raspberry Pi

L'ADC que nous avons sélectionné possède une interface de communication SPI. Le SPI est un bus de communication full-duplex, fonctionnant selon le mode maître-esclave. L'horloge de notre communication étant fourni par la Raspberry Pi, c'est cette dernière qui joue le rôle de maître.

Communication Bluetooth Raspberry PI-Smartphone

Afin de pouvoir envoyer la valeur mesurée au smartphone, nous utilisons une communication Bluetooth. Sur la RPI 3, le Bluetooth est natif, nous n'avons pas besoin d'une clef Bluetooth. La communication fonctionne sur le principe de sockets.

Dans notre système, la RPI joue le rôle de serveur. La RPI attend qu'un client se connecte. Pour cela, le client se connecte via l'UUID de la RPI. L'UUID est un identifiant unique permettant l'identification d'un système communicant. Une fois connecté, un socket est créé et ouvert, puis la RPI attend que le client lui envoie une requête. Voir la partie Application Android pour la liste de requêtes recevables.

Une fois la communication terminée, le socket se ferme et est détruite.

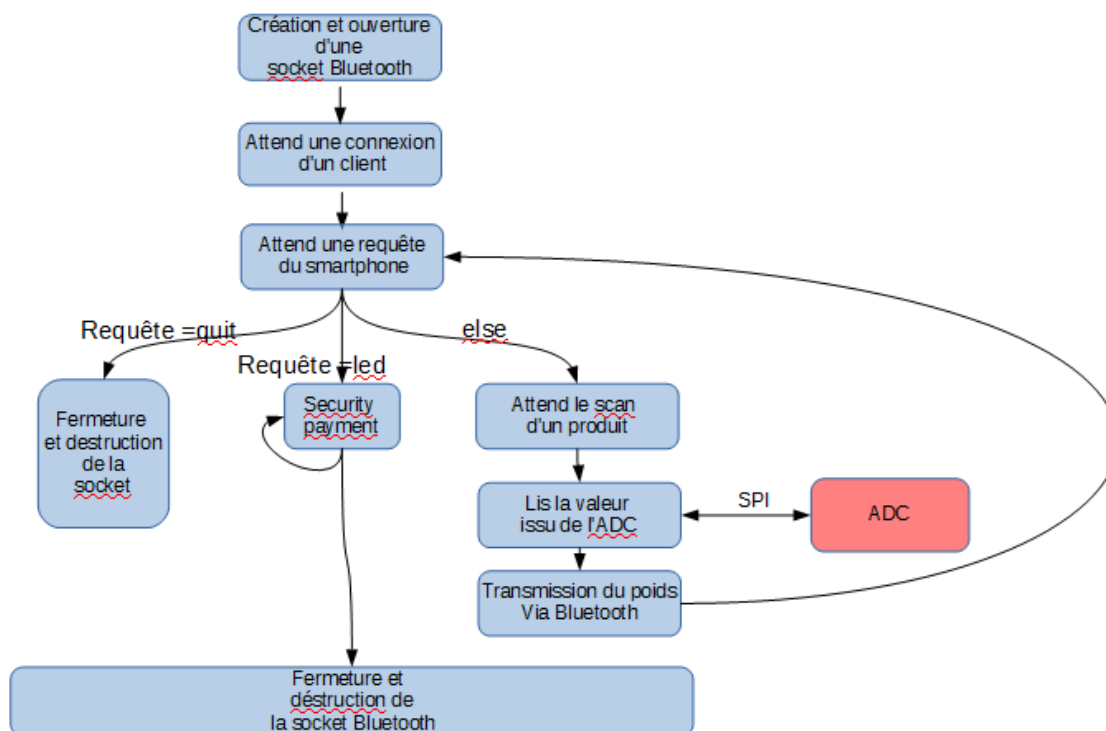


Figure 7: Fonctionnement Software de la Balance (Bluetooth) (Serial Peripheral Interface (SPI)) (Python bluetooth.BluetoothSocket Examples).

4. Suivi d'utilisateur

Nous avons réfléchi aux différentes manières d'établir un suivi du client en utilisant les différents outils à notre disposition. Le suivi demande de réfléchir à 2 problématiques :

La première concerne la mécanique/électronique du système, quels composants sont nécessaires et comment les câbler pour que le robot suive l'utilisateur ?

La seconde est la problématique logicielle, quels algorithmes développer pour que la Raspberry dirige les moteurs grâce aux informations des capteurs ?

Nous avons donc cherché à répondre à ces problématiques au cours des mois durant lesquels nous avons pu travailler le projet.

a. Montage du chariot

En ce qui concerne les moteurs, nous avons été confrontés à différentes problématiques, il a fallu les fixer au chariot, et grâce à une chaîne à maillons de type 06B entraîner les roues arrière indépendamment. Les roues ont-elles aussi dû être changées pour d'autres plus maniables. Enfin, des pignons à dents 06B ont été vissés à la roue et au moteur pour être compatibles à la chaîne les reliant.

Nous avons utilisé des pignons de 45 et 115 mm pour respectivement pour le moteur et la roue, cela permet de diminuer la vitesse et d'augmenter le couple du moteur.

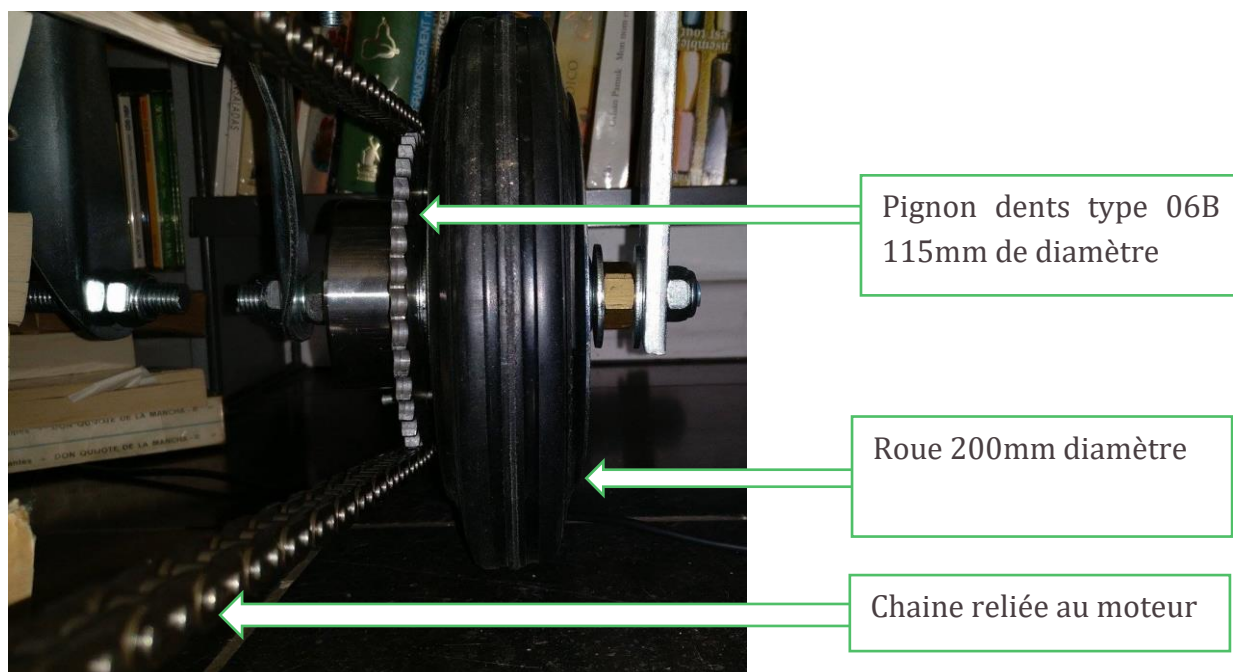


Figure 8: Schéma du montage d'une roue arrière du chariot

L'ensemble est fixé sous une planche en bois par mesure de sécurité.

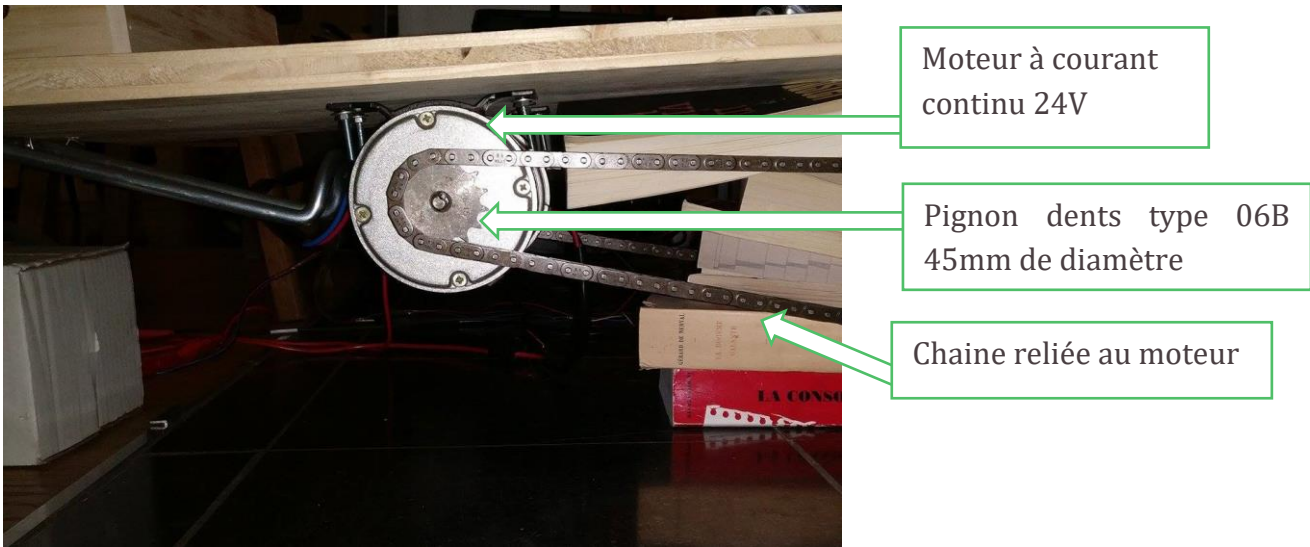


Figure 9:Photo de montage d'un moteur à courant continu

Après avoir monté les moteurs, nous nous sommes occupés de la partie électronique du système, il s'agissait pour nous d'asservir ces moteurs avec la Raspberry, tout en les alimentant avec des batteries, et de brancher les sonars.

Nous avons choisi de diriger chaque moteur avec le Mosfet Canal N, 343A 40V IRLB3034PbF commandé par la Raspberry. Il est capable de supporter la puissance transmise par les batteries, il passe à l'état haut pour $V_{gs} = 2.5V$, et à l'état bas pour $V_{gs} = 1V$, avec V_{gs} la tension entre la gate G (commandée par la Raspberry) et la source S qui est pour nous reliée à la masse. Ceci permet de contrôler le Mosfet avec les ports GPIO de la Raspberry en pwm, ceux-ci délivrent une tension allant jusqu'à 3.3V.

Nous avons appliqué un pont diviseur de tension entre La pin GPIO et le transistor par sécurité :

$$\text{Ainsi } V_{gs} = V_{gpio} \frac{R_1}{R_1 + R_2} = 3.3 \frac{10000}{10000 + 1000} = 3V$$

Voici le montage que nous avons effectué :

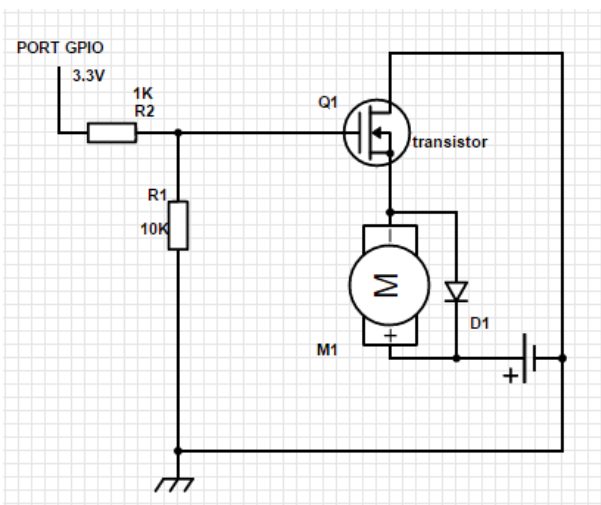


Figure 10 : Schéma de câblage d'un moteur

Concernant les capteurs, nous avons mis en place un HC-SR04. C'est un capteur à ultrasons utilisé pour mesurer la distance, il émet une onde et calcule le temps qu'elle met à revenir pour déterminer la distance qu'elle a parcouru. Nous nous en servons pour détecter les obstacles (Marsh, 2015).

Nous avons réalisé un montage avec les 4 broches disponibles sur l'appareil :

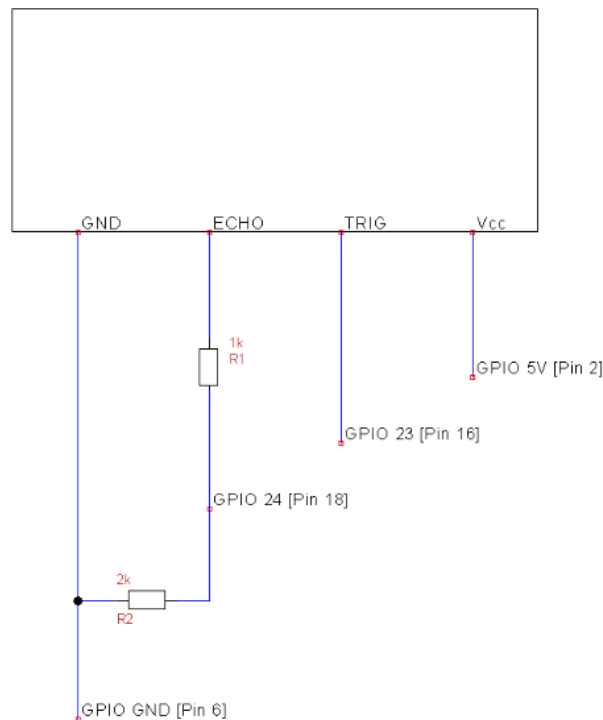


Figure 11: montage d'un capteur à ultrasons HC-SR04

b. Bluetooth

BLE

Après les nombreuses recherches et comparaisons des différentes technologies existantes pour la transmission de données sans fil, nous avons décidé d'utiliser le BLE. Le BLE (Bluetooth basse consommation) permet un transfert de données plus rapide avec une faible consommation, il est de plus disponible sur la grande majorité des smartphones actuels. Il nécessite cependant la mise en place d'une relation maître-esclave pouvant être contraignante.

RSSI

Pour le suivi d'utilisateur, nous avons utilisé la triangulation qui est une technique permettant de déterminer la position d'un point en mesurant les angles entre ce point et d'autres points de référence dont la position est connue, et ceci plutôt que de mesurer directement la distance entre les points (Triangulation). Pour cela, nous avons décidé de récupérer le RSSI d'un téléphone.

RSSI signifie Received Signal Strength Indication, cela permet de mesurer la puissance en réception d'un signal reçu émis par une antenne.

Nous allons utiliser le RSSI pour calculer la distance entre l'utilisateur et le chariot, dans le but que le chariot puisse suivre l'utilisateur sans le perdre. Le calcul de distance est basé sur l'intensité du signal reçu, si l'intensité du signal est faible, c'est-à-dire que le signal a diminué, alors le chariot approche l'utilisateur ; si l'intensité du signal est forte, alors le chariot garde la même distance.

La valeur de RSSI varie de 0dbm à 120dbm, 0dbm est le signal le plus fort, donc le signal est plus fort quand il approche de 0dbm, et il est plus faible quand il approche de 120dbm.

Pour calculer le RSSI, nous allons utiliser $d=10^{((ABS(RSSI)-A)/(10*n))}$

A : L'intensité du signal à 1 mètre en dBm

d : Distance en mètre

n : L'exposant de path-loss

Nous avons utilisé la Raspberry pour récupérer la valeur de RSSI du smartphone de l'utilisateur. Les difficultés pour la récupération de RSSI sont de choisir le bon algorithme et de l'automatiser.

c. Solutions logicielles

Notre objectif était d'offrir une solution tirant au maximum partie des outils mis à disposition et d'utiliser les connaissances acquises au cours de l'année. Nous avons décidé de développer un algorithme de Reinforcement learning, plus précisément de Q-learning pour le suivi d'utilisateur. Il s'agit d'une branche du Machine-learning dans laquelle le robot cherche le meilleur chemin pour accéder à un état idéal à partir d'une position inconnue. Dans ce type d'algorithme des « états » sont atteints en effectuant des « actions », à chaque état le robot calcule quelle action effectuer pour atteindre le prochain état grâce à un système de « Qvalues ». Les values sont calculées en fonction de la « récompense » que peut offrir chaque action. Les récompenses sont définies arbitrairement en fonction de l'objectif du robot.

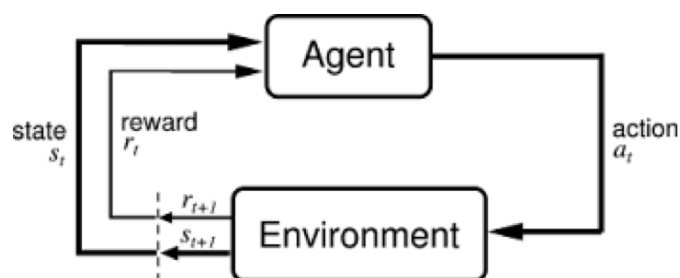


Figure 12 : Fonctionnement d'un algorithme de Q-learning

La mise en place de type d'algorithme se fait en 2 phases, une première est l'entraînement, durant celle-ci le robot va tester toutes les possibilités et calculer les Qvalues permettant d'accéder à l'état optimal. En seront extraites et sauvegardées les Qvalues de toutes les actions possibles à chaque état.

Durant la seconde phase le robot connaît le chemin optimal et ne fait que déterminer les actions à effectuer en fonction de son état et des Qvalues déterminées lors de la première phase.

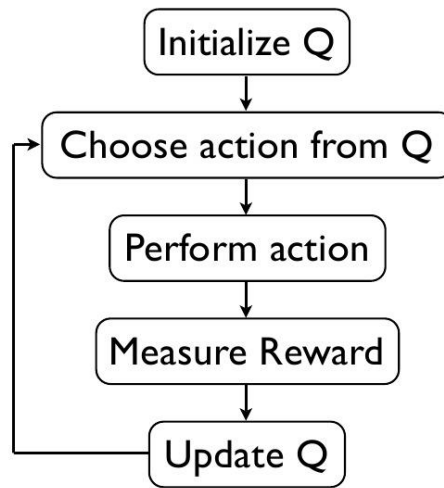


Figure 13 : Phase d'apprentissage du robot

Ici nos états sont les états des capteurs, pour que le robot détecte parfaitement le client nous implémentons 4 capteurs ultrason dans l'algorithme, ce chiffre peut être facilement augmenté ou diminué en fonction des besoins réels. Nous avons également 3 capteurs Bluetooth permettant pour faire de la triangulation, chaque capteur renvoie sa distance par rapport au smartphone, de cette manière nous pouvons déduire la position de l'utilisateur par rapport au chariot. Nous avons donc au total 7 capteurs servant à définir les états.

Concernant les actions, nous en avons défini 4 : avancer – tourner à droite – tourner à gauche – s'arrêter. Ces dernières se matérialisent par la variation de la tension envoyée aux moteurs.

Ce type d'algorithme demande cependant beaucoup d'itérations, il nous est impossible d'entraîner le robot entièrement en conditions réelles pour des raisons de temps, de plus nous ne disposons pas de chargeur pour les batteries alimentant les moteurs. C'est pourquoi nous avons décidé de créer une simulation permettant au robot d'apprendre les différentes actions à effectuer en fonction des états. Les résultats de cet entraînement pourront être transférés au robot.

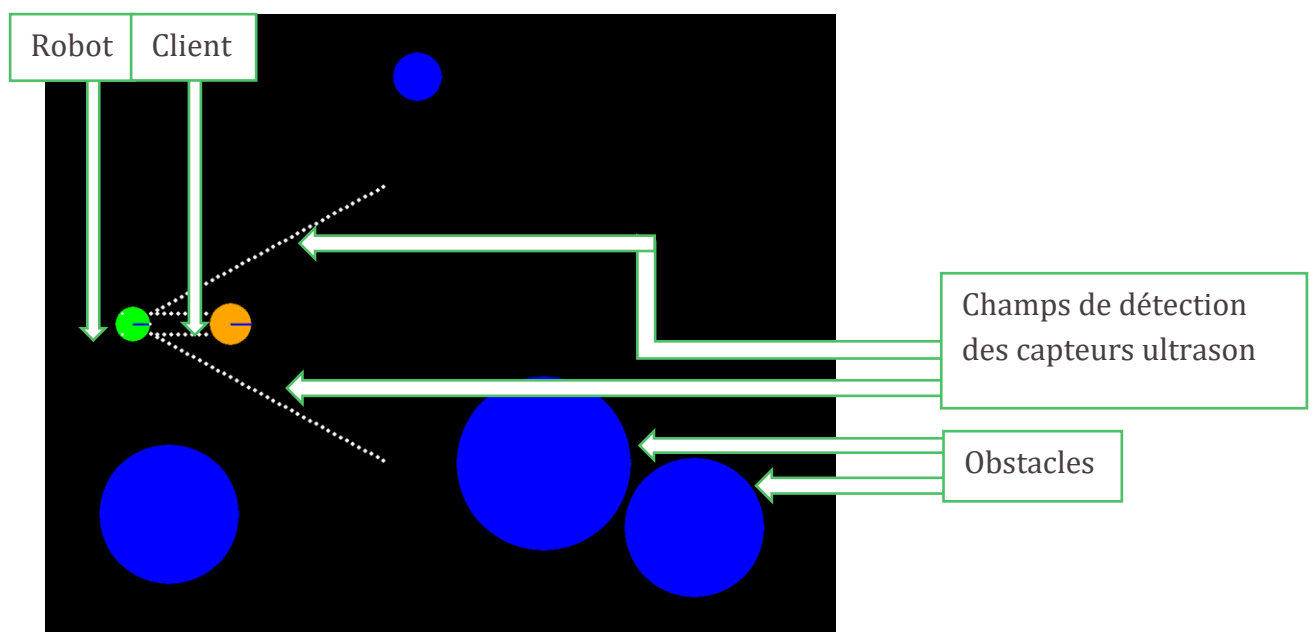


Figure 14 : Simulation pour la phase d'apprentissage du robot

Nous utilisons les bibliothèques pygame et pymunk pour générer l'environnement du robot dans la simulation. Ces bibliothèques proposent des outils complets pour afficher des formes, les déplacer dans un environnement 2D.

Nous utilisons la bibliothèque keras pour prédire les valeurs, accéder au tableau des actions/états. Cela nous permet d'enregistrer l'ensemble des valeurs nécessaires à la phase 2 dans un format compressé .h5, et sauver de l'espace mémoire.

Après avoir mis en place la simulation, nous avons défini les fonctions servant à l'apprentissage. A chaque étape, le robot devra calculer les variables grâce aux actions suivantes :

- read0-read1-read2-read3 : gaussienne appliquée aux distances pour avoir une valeur importante lorsque l'utilisateur est à 1m, faible sinon.
- Récompense(reward) : Elle détermine la valeur d'un état

$$R = \frac{(2000 \cdot \text{read0} + 6000 \cdot \text{read1} + 6000 \cdot \text{read2} + 2000 \cdot \text{read3} + \sum(\text{ble_distances}))}{10}$$

- Ble_distances : ensemble des distances mesurées par les 3 capteurs Bluetooth.
- Qvalue : sert à déterminer la valeur de chaque action possible en fonction de la reward de l'état actuel et des Qvalues des prochains états (liés chacun à une action possédant une Qvalue)

$$Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \gamma \cdot \text{Max}[Q(\text{next state}, \text{all actions})]$$

A chaque itération le robot calcule ces variables et la Qvalue de chaque action pour l'état où il se trouve, puis se déplace vers l'état suivant. Nous faisons 100000 itérations pour nous assurer de la convergence de l'algorithme.

Après avoir généré toutes les Qvalues pour les actions possibles à chaque état, nous réalisons l'algorithme de la phase 2. Il s'agit ici de déplacer le chariot en fonction des Qvalues des états. A chaque itération, le robot analyse son état grâce aux valeurs des 7 capteurs. Il regarde ensuite quelle est la Qvalue maximale pour cet état, il effectue l'action correspondant à cette Qvalue maximale.

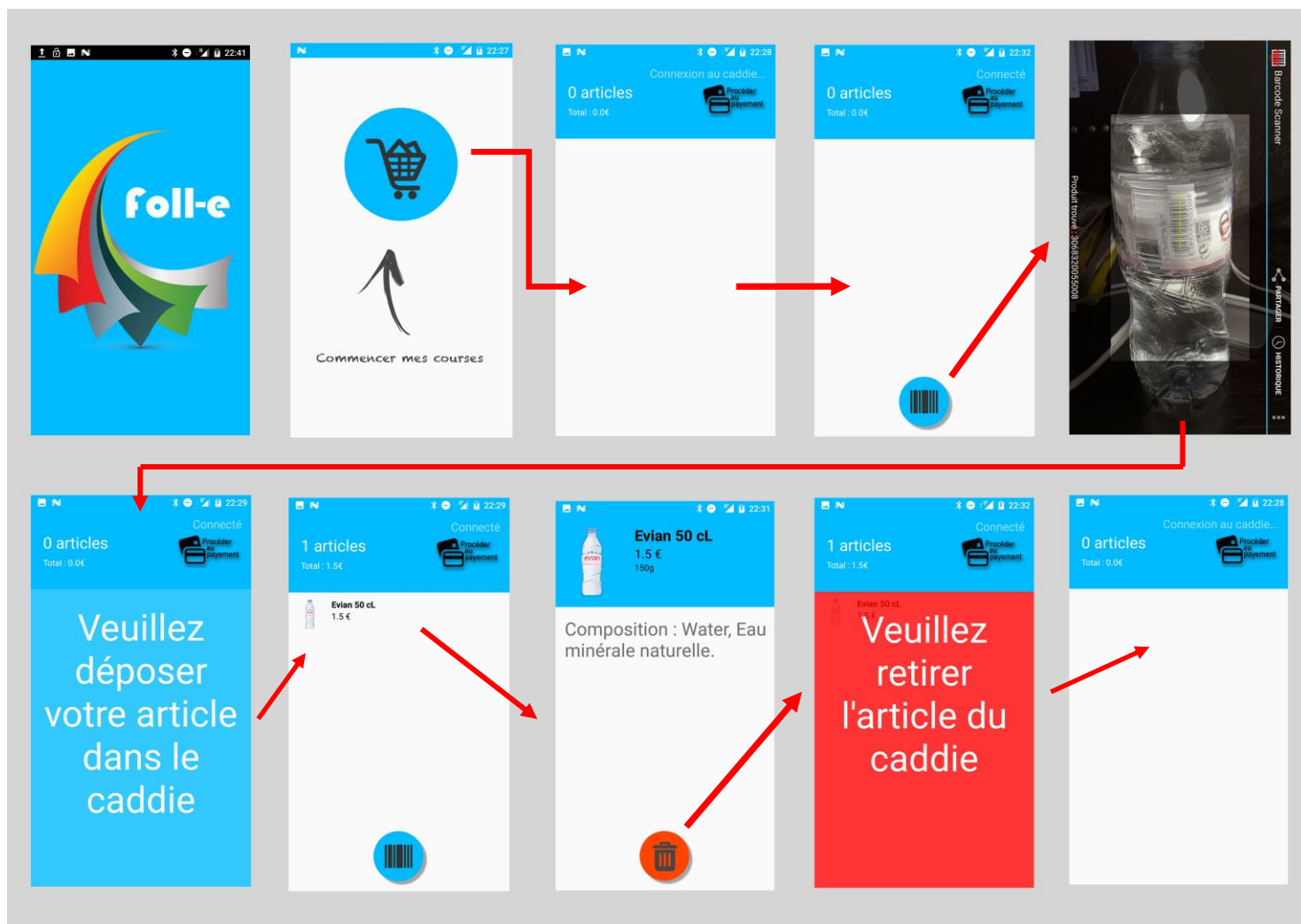
I. RESULTATS

1. Résultats obtenus

a. Application

Voici le résultat final de l'application :

Partie Scan et interaction avec les produits/caddie :



Lorsque l'application se lance, on arrive sur un écran d'accueil composé d'un bouton permettant de commencer ses courses. Sur un appui de ce bouton, nous nous retrouvons sur l'activité principale de l'application. Cette activité est composée d'un panneau d'information donnant des indications sur l'état actuel de la connexion avec le caddie (connecté, en connexion, erreur de connexion), le nombre d'articles présents dans le caddie, le prix total des courses ainsi qu'un bouton pour procéder au paiement.

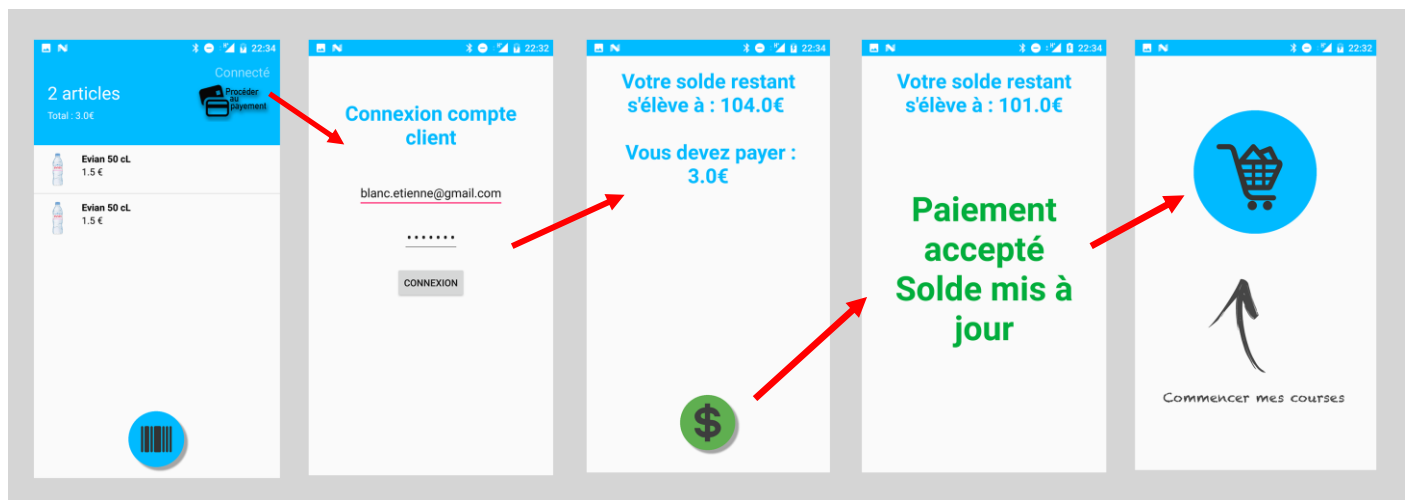
Lorsque la connexion est établie, un bouton scan apparaît. Sur un appui sur ce bouton, l'application nous redirige vers l'activité scan, permettant de scanner un article via l'appareil photo. Lorsque le scan est effectué, un message nous indique qu'il faut déposer l'article dans le caddie. L'application a

envoyé en arrière-plan, l'ordre au caddie de retourner le poids mesuré. Tant que le poids attendu n'est pas correct, l'application n'ajoute pas l'article. Lorsque le poids reçu correspond bien au poids théorique du caddie, l'article est alors ajouté à la liste des articles du caddie. Le panneau d'affichage se met alors à jour.

Un appui sur un article de la liste nous permet d'obtenir plus d'informations sur celui-ci, comme la liste de ses ingrédients, taux d'acides gras, ...

Pour supprimer un article de la liste, il suffit d'appuyer sur le bouton corbeille présent sur la fiche du produit à supprimer. Après appui, nous revenons sur l'activité principal indiquant au client de retirer l'article en question du caddie. Tant que le poids du caddie ne correspond pas au poids qu'il devrait contenir après l'enlèvement de l'article, le client ne peut plus interagir avec l'activité.

Partie Paiement :



Lorsque l'utilisateur a fini ses courses, pour procéder au paiement, il suffit d'appuyer sur le bouton correspondant. L'application nous amène vers une page d'authentification permettant de se connecter à son compte. Après la connexion, on peut prendre connaissance, du solde de son compte et ainsi payer en appuyant sur le bouton \$. Après l'étape de paiement, l'application nous redirige vers l'accueil permettant d'effectuer de nouvelles courses.

b. Balance

Les résultats de la balance concernant la valeur que l'on récupère en sortie de la chaîne d'acquisition :

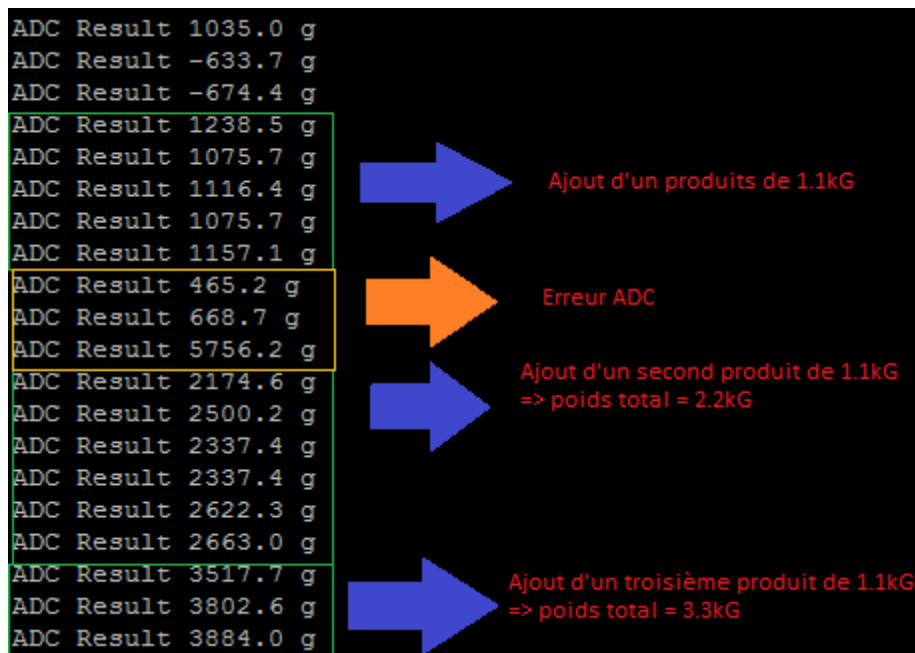


Figure 15 : Les résultats de la balance concernant la valeur que l'on récupère en sortie de la chaîne d'acquisition

Nous avons calculé le taux d'erreur de notre système.

Poids réel (g)	Poids mesuré (g)	Taux d'erreur
0	250	NULL
1100	1015	8%
2200	2400	10%
3300	3800	15%
6600	5800	12.1%

Nous avons obtenus une moyenne de 11.275% d'erreur sur notre valeur mesurée. En dessous de 1kG, il est très difficile de détecter une variation significative.

De plus nous avons constaté qu'en fonction de la position du produit sur la balance, le poids récupéré n'était pas le même. Cela complique notre traitement du poids via la Raspberry.

c. Bluetooth

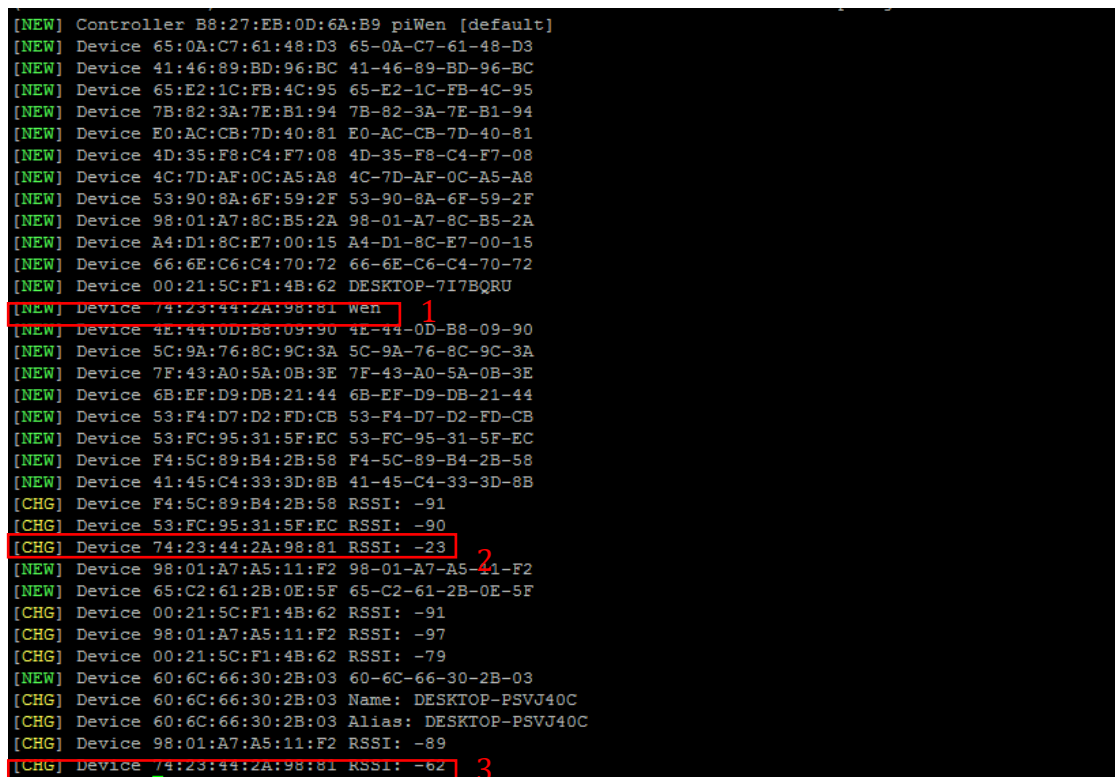
Programme

Pour le suivi d'utilisateur, nous avons utilisé la Raspberry pi dans le but pour récupérer la valeur RSSI d'un mobile. Pour cela, nous avons programmé en bash :

```
#!/bin/bash
# enable bluetooth
sudo systemctl start bluetooth
bluetoothctl
```

Résultats

D'après le programme précédent, nous avons obtenu :



```
[NEW] Controller B8:27:EB:0D:6A:B9 piWen [default]
[NEW] Device 65:0A:C7:61:48:D3 65-0A-C7-61-48-D3
[NEW] Device 41:46:89:BD:96:BC 41-46-89-BD-96-BC
[NEW] Device 65:E2:1C:FB:4C:95 65-E2-1C-FB-4C-95
[NEW] Device 7B:82:3A:7E:B1:94 7B-82-3A-7E-B1-94
[NEW] Device E0:AC:CB:7D:40:81 E0-AC-CB-7D-40-81
[NEW] Device 4D:35:F8:C4:F7:08 4D-35-F8-C4-F7-08
[NEW] Device 4C:7D:AF:0C:A5:A8 4C-7D-AF-0C-A5-A8
[NEW] Device 53:90:8A:6F:59:2F 53-90-8A-6F-59-2F
[NEW] Device 98:01:A7:8C:B5:2A 98-01-A7-8C-B5-2A
[NEW] Device A4:D1:8C:E7:00:15 A4-D1-8C-E7-00-15
[NEW] Device 66:6E:C6:C4:70:72 66-6E-C6-C4-70-72
[NEW] Device 00:21:5C:F1:4B:62 DESKTOP-7I7BQRU
[NEW] Device 74:23:44:2A:98:81 wen 1
[NEW] Device 4E:44:0D:B8:09:90 4E-44-0D-B8-09-90
[NEW] Device 5C:9A:76:8C:9C:3A 5C-9A-76-8C-9C-3A
[NEW] Device 7F:43:A0:5A:0B:3E 7F-43-A0-5A-0B-3E
[NEW] Device 6B:EF:D9:DB:21:44 6B-EF-D9-DB-21-44
[NEW] Device 53:F4:D7:D2:FD:CB 53-F4-D7-D2-FD-CB
[NEW] Device 53:FC:95:31:5F:EC 53-FC-95-31-5F-EC
[NEW] Device F4:5C:89:B4:2B:58 F4-5C-89-B4-2B-58
[NEW] Device 41:45:C4:33:3D:8B 41-45-C4-33-3D-8B
[CHG] Device F4:5C:89:B4:2B:58 RSSI: -91
[CHG] Device 53:FC:95:31:5F:EC RSSI: -90
[CHG] Device 74:23:44:2A:98:81 RSSI: -23 2
[NEW] Device 98:01:A7:A5:11:F2 98-01-A7-A5-41-F2
[NEW] Device 65:C2:61:2B:0E:5F 65-C2-61-2B-0E-5F
[CHG] Device 00:21:5C:F1:4B:62 RSSI: -91
[CHG] Device 98:01:A7:A5:11:F2 RSSI: -97
[CHG] Device 00:21:5C:F1:4B:62 RSSI: -79
[NEW] Device 60:6C:66:30:2B:03 60-6C-66-30-2B-03
[CHG] Device 60:6C:66:30:2B:03 Name: DESKTOP-PSVJ40C
[CHG] Device 60:6C:66:30:2B:03 Alias: DESKTOP-PSVJ40C
[CHG] Device 98:01:A7:A5:11:F2 RSSI: -89
[CHG] Device 74:23:44:2A:98:81 RSSI: -62 3
```

Figure 16: Résultat obtenu de RSSI

Dans le premier cadre : Affichage de l'adresse Mac

Dans le deuxième cadre : Récupérer la RSSI quand le portable est proche de la Raspberry

Dans le troisième cadre : Récupérer la RSSI quand le portable est loin de la Raspberry

Courbe

Pour avoir des variables pour fiables, nous avons fait un étalonnage. Nous avons récupéré les valeurs RSSI avec des différentes distances :

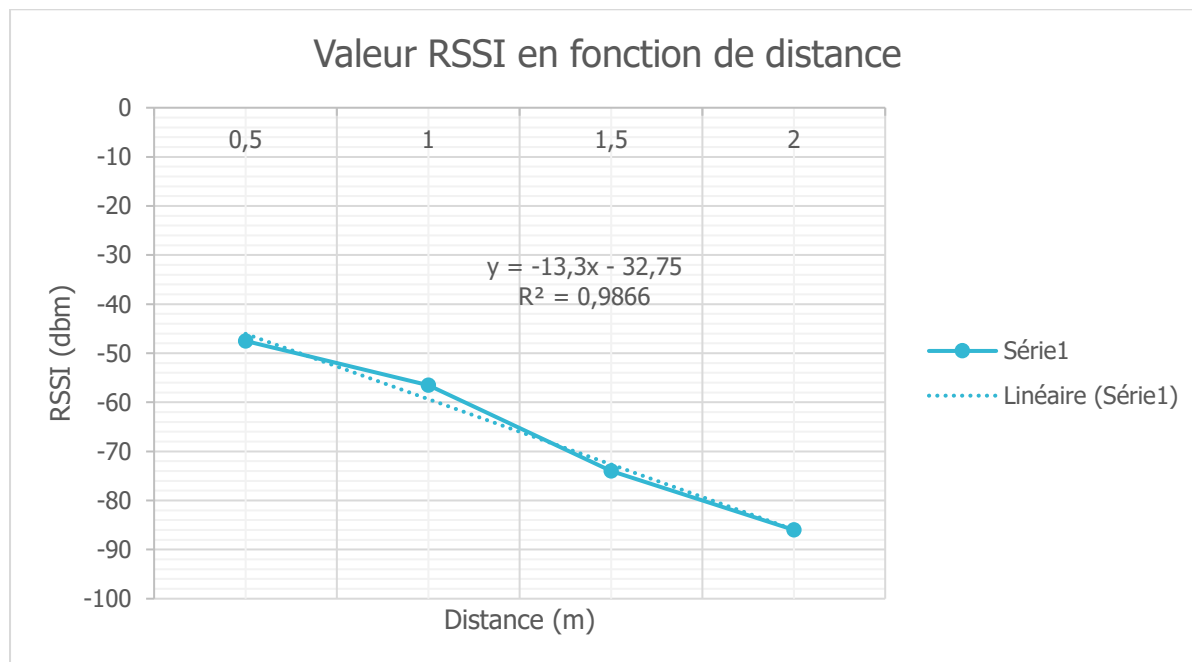


Figure 17: Courbe représente la valeur RSSI en fonction de distance

D'après la courbe, nous observons que la valeur RSSI récupéré par la Raspberry Pi qui est assez fiable. R^2 est proche de 0

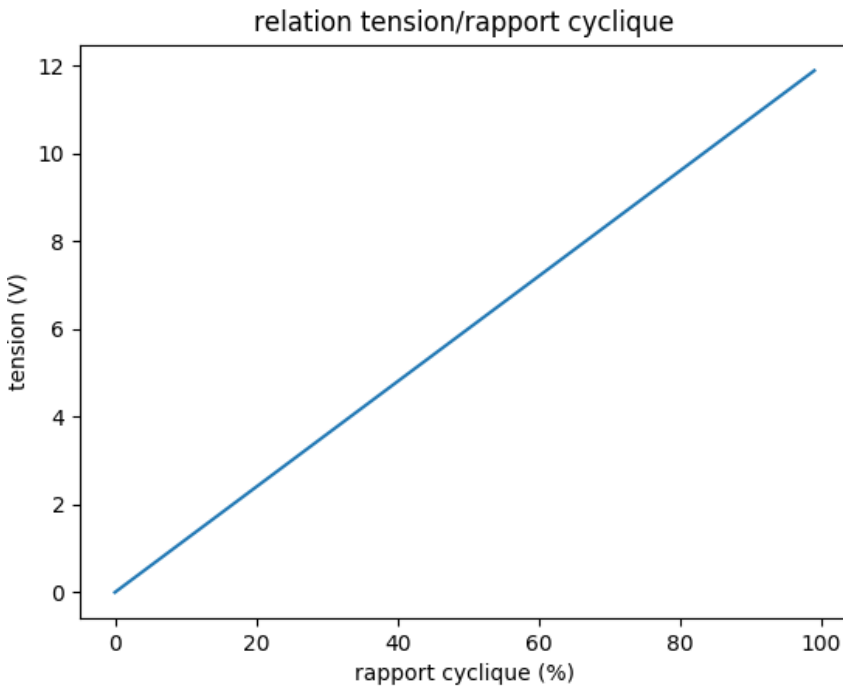
d. Capteurs ultrason

Concernant les capteurs à ultrason, après avoir réalisé le câblage, ils sont capables de renvoyer la distance exacte au centimètre près. Cependant, des erreurs peuvent survenir si le capteur est déplacé brusquement.

Distance	0 à 3m	3m et plus
Fiabilité	Fiable au centimètre près	Distance mesurée 3200cm

e. Moteurs

Nous réussissons à asservir les moteurs à l'aide des transistors reliés à la Raspberry, Nous rendons le transistor passant ou non via un port GPIO de la Raspberry en mode pwm, cela permet de changer l'état du transistor à une certaine fréquence pour monter ou baisser la vitesse des moteurs. Le rapport cyclique de la pwm indique le temps durant lequel le signal est à l'état haut par rapport au temps à l'état bas. C'est en faisant varier ce dernier que nous pouvons réguler la vitesse des moteurs.



Nous faisons tourner les moteurs en définissant la pwm sur 50%, ainsi ils seront alimentés à 6V, c'est la valeur minimale pour faire avancer le système.

Pour arrêter les moteurs, nous définissons la pwm sur 0%, pour que le transistor soit bloquant et que les moteurs ne tournent pas.

Asservissement des moteurs à partir de capteurs

Pour vérifier que l'ensemble du système est cohérent et qu'il est possible d'utiliser les capteurs à ultrason pour commander les moteurs, nous avons développé un programme qui arrête les moteurs si un capteur détecte une personne à moins d'un mètre et avance si une personne est détectée au-delà.

```
pi@raspberrypi: ~/Folle/Folle_ML
File Edit Tabs Help
g anyway. Use GPIO.setwarnings(False) to disable warnings.
GPIO.setup(LedMotdroit, GPIO.OUT) # Set LedMotdroit's mode is output
asset1Cap_2mot.py:104: RuntimeWarning: This channel is already in use, continuin
g anyway. Use GPIO.setwarnings(False) to disable warnings.
GPIO.setup(LedMotGauche, GPIO.OUT) # Set LedMotdroit's mode is output
Distance Measurement In Progress
Waiting For Sensor To Settle
duty cycle : 1
duty cycle : 1
DC 1 : 1
DC 2 : 1
duty cycle : 1
duty cycle : 1
DC 1 : 1
DC 2 : 1
Distance: [13.72, -1, -1, -1]
Distance Measurement In Progress
Waiting For Sensor To Settle
duty cycle : 1duty cycle : 1
DC 1 : 1
DC 2 : 1
duty cycle : 1
duty cycle : 1
DC 1 : 1
DC 2 : 1
Distance: [96.42, -1, -1, -1]
Distance Measurement In Progress
Waiting For Sensor To Settle
duty cycle : 50
duty cycle : 50
DC 2 : 50
DC 1 : 50
duty cycle : 50
duty cycle : 50
DC DC 1 2 : 50
: 50
Distance: [107.43, -1, -1, -1]
Distance Measurement In Progress
Waiting For Sensor To Settle
duty cycle : 50
duty cycle : 50
DC DC 1 2 : 50
: 50
Distance: [6.93, -1, -1, -1]
Distance Measurement In Progress
Waiting For Sensor To Settle
duty cycle : 1
duty cycle : 1
DC DC 1 2 : 1
: 1
^Z
[2]+ Stopped sudo python asset1Cap_2mot.py
pi@raspberrypi:~/Folle/Folle_ML $ scrot
```

Capteur retourne 13,7cm, objet proche détecté

Rapport cyclique de la pwm faible pour arrêter les moteurs

Capteur retourne 107,4cm, objet éloigné détecté

Rapport cyclique de la pwm 50% pour se rapprocher

Figure 18 : Résultats obtenu lors de la commande des moteurs avec le capteur ultrason

Nous voyons sur l'image ci-dessus que la pwm est à 0(moteurs arrêtés) lorsqu'un objet à moins de 1m et à 50 lorsqu'un objet est à plus de 1m.

Nous avons donc asservi avec succès les moteurs et faisons avancer un objet ou pas selon la valeur détectée par un capteur.

f. Machine Learning

Du point de vue de l'algorithme de Q-learning, nous avons réussi à faire converger le robot, il est capable de localiser l'utilisateur grâce à la triangulation et éviter les obstacles grâce aux capteurs ultrason. L'apprentissage a été efficace, les différentes actions possibles ont été évaluées en fonction des différents états.

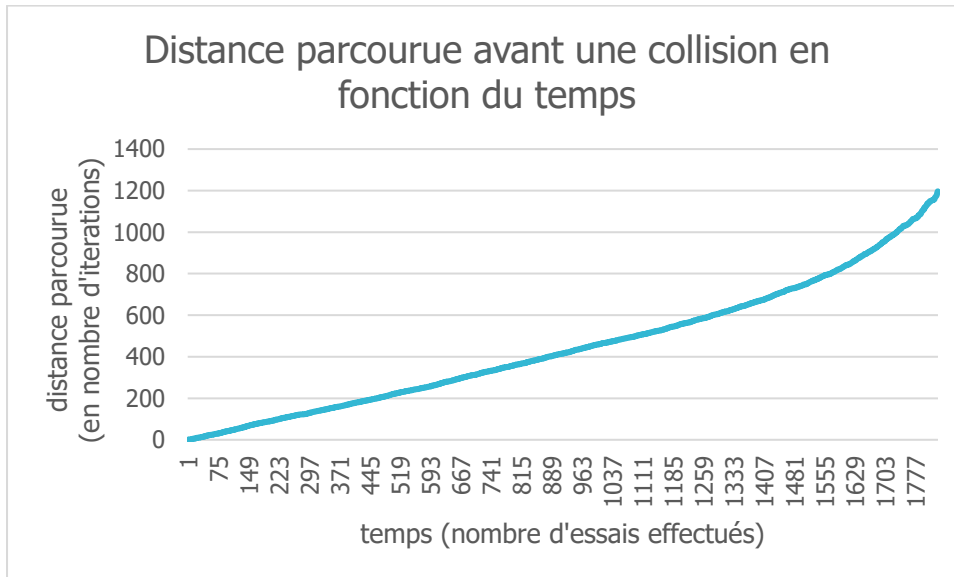


Figure 19 : courbe montrant le nombre d'itérations effectuées par le robot avant de rencontrer un obstacle durant l'apprentissage en fonction du temps

Une itération correspond à un cycle « acquisition de l'état grâce aux données des capteurs/calculs des Qvalues de l'état actuel/exécution de la prochaine action ». Le temps est compté en nombre d'essais effectués, un essai démarre avec l'initialisation du robot et se termine avec une collision.

Cette courbe montre que le robot réussit à chaque essai à faire un nombre d'itérations plus important avant d'avoir une collision.

Nous avons ensuite développé une simulation pour tester notre robot, il est possible de déplacer nous même le client à l'aide des touches du clavier pour vérifier que le robot essaie de nous suivre.

Cette dernière fonctionne, le robot suit le client tout en s'arrêtant lorsqu'il détecte un obstacle.

L'algorithme est prêt à être implémenté sur la Raspberry, mais le dispositif bluetooth n'étant pas près, il est impossible de localiser le client et donc de le différencier d'un obstacle.

2. Discussion sur les résultats

Côté application, nous avons répondu à toutes les exigences principales demandées par le cahier des charges. L'application permet donc de scanner un produit, d'obtenir des informations dessus tel que son poids, prix, ingrédients... tout cela en contrôlant l'ajout et l'enlèvement d'article du caddie, permettant ainsi d'éviter les erreurs et les vols. Enfin nous avons réalisé avec succès le système de paiement, qui fonctionne avec un compte client. Chaque client possède un compte qu'il peut créditer. Il peut ainsi faire ses courses dans les magasins utilisant le système de chariot Foll-E. A l'étape paiement, le compte de l'utilisateur est débité du montant de ses courses.

Les parties théorique et logicielle de suivi d'utilisateur sont au point, il reste à mettre en place le Bluetooth pour appliquer l'algorithme de suivi dans la réalité. Sur la partie mécanique et électronique, les capteurs ultrason fonctionnent et sont relativement fiables, ils servent aujourd'hui à asservir les moteurs. Ces derniers sont eux aussi fixés et en état de faire avancer le chariot. Il reste à tester le système en conditions réelles, pour voir les dangers possibles liés au suivi et les anticiper.

Du point de vue de la balance, la communication Bluetooth fonctionne bien. Néanmoins à la vue des résultats concernant la précision du système, il faudrait pouvoir augmenter la précision. En plus de revoir la mécanique de la balance, il faudrait ajouter d'autres jauges de contrainte afin de pouvoir détecter une variation du poids pour n'importe quelle position du produit dans le caddie.

CONCLUSION

Travailler pour le projet Foll-E a été l'occasion pour nous de réaliser un produit se voulant proche des besoins actuels de l'industrie des supermarchés. Nous avons imaginé un chariot autonome, qui facilite la vie des consommateurs. Nous l'avons pensé pour permettre au consommateur de payer ses achats depuis son téléphone sans avoir à passer en caisse, et avons voulu qu'il puisse suivre son utilisateur afin de l'aider dans ses achats.

Au cours de l'année nous avons analysé les solutions aux différentes problématiques du projet. Nous nous sommes réparti les tâches en fonction des compétences de chaque membre de l'équipe et des objectifs à accomplir.

Chacun des 5 membres a apporté sa solution pour la mise en place de Foll-E. Etienne a développé le scan de code barre, le paiement, la partie serveur middleware et base de données. Quentin a mis en place le protocole Bluetooth entre la Raspberry et l'application, a réalisé l'architecture de l'application et la vérification des articles placés dans le caddie. Wen a travaillé sur les capteurs pour la solution de suivi et la communication de groupe. Bastien a réalisé le montage de la balance, et le script de communication avec l'application. Adrien a pour sa part conçu l'algorithme de Q-learning pour le suivi d'utilisateur, monté le chariot en plus de son rôle de chef de projet.

Tous les membres ont eu l'occasion de travailler en synergie avec leurs homologues à plusieurs reprises au long de l'année.

La balance est fonctionnelle, il est possible de rajouter, supprimer des articles au chariot et payer. L'algorithme de suivi est au point et attend d'être testé en conditions réelles. Enfin, la partie hardware du chariot est fonctionnelle et montée.

Il reste donc à finaliser la triangulation Bluetooth, l'algorithme de Q-learning peut être perfectionné pour mieux réagir aux différentes situations. La pesée peut être améliorée en consolidant la balance, l'application peut être maintenue et des fonctionnalités peuvent être ajoutée. Ces extensions peuvent être développées lors d'un prochain PPE ou PFE.

BIBLIOGRAPHIE

- BARBOT, H. H.-E.-J.-P. (2015, 06 18). *Capteurs et chaîne d'acquisition*. Récupéré sur Eduscol:
<http://eduscol.education.fr/sti/sites/eduscol.education.fr/sti/files/ressources/pedagogiques/6151/6151-capteur-et-chaine-dacquisition-ens.pdf>
- Bluetooth*. (s.d.). Récupéré sur www.commentcamarche.net:
<http://www.commentcamarche.net/contents/108-bluetooth-comment-ca-marche>
- Marsh, J. (2015, 06 30). *Ultrasonic rangefinder*. Récupéré sur Développez: <https://raspberrypi.developpez.com/cours-tutoriels/capteur/mag-pi-utiliser-port-gpio/partie-3-telemetre-ultrason/>
- Python bluetooth.BluetoothSocket Examples*. (s.d.). Récupéré sur programcreek:
<http://www.programcreek.com/python/example/14725/bluetooth.BluetoothSocket>
- Serial Peripheral Interface (SPI)*. (s.d.). Récupéré sur Sparkfun:
<https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>
- Teknomo, K. (2005). *Q-Learning Numerical Example*. Récupéré sur Revoledu:
<http://people.revoledu.com/kardi/tutorial/ReinforcementLearning/Q-Learning-Example.htm>
- Triangulation*. (s.d.). Récupéré sur Wikipédia: <https://fr.wikipedia.org/wiki/Triangulation>

Liste exhaustive des produits du projet accessibles

- <https://github.com/notetienne/folle>
- https://github.com/adrienchevrier/Folle_ML

TABLE DES FIGURES

Figure 1: Répartition des tâches.....	6
Figure 2: L'avancement du projet.....	6
Figure 3: Fonctionnement de l'échange de données entre l'application et le serveur.....	8
Figure 4: Caractérisation du capteur.....	9
Figure 5: Synoptique de la Balance (BARBOT, 2015).	9
Figure 6: Système de la Balance.	10
Figure 7: Fonctionnement Software de la Balance (Bluetooth) (Serial Peripheral Interface (SPI)) (Python bluetooth.BluetoothSocket Examples).....	11
Figure 8: Schéma du montage d'une roue arrière du chariot	12
Figure 9: Photo de montage d'un moteur à courant continu	13
Figure 10 : Schéma de câblage d'un moteur	13
Figure 11: montage d'un capteur à ultrasons HC-SR04.....	14
Figure 12 : Fonctionnement d'un algorithme de Q-learning.....	15
Figure 13 : Phase d'apprentissage du robot	16
Figure 14 : Simulation pour la phase d'apprentissage du robot.....	16
Figure 15 : Les résultats de la balance concernant la valeur que l'on récupère en sortie de la chaîne d'acquisition	20
Figure 16: Résultat obtenu de RSSI.....	21
Figure 17: Courbe représente la valeur RSSI en fonction de distance.....	22
Figure 18 : Résultats obtenu lors de la commande des moteurs avec le capteur ultrason	24
Figure 19 : courbe montrant le nombre d'itérations effectuées par le robot avant de rencontrer un obstacle durant l'apprentissage en fonction du temps	25