

**Nano Ordinateurs  
Gestion de projet   
Robot Baliseur**

Mars 2018

Version 1.0

Wattin Jérôme  
Sam Mahaux

Fabio Cumbo

Timothée Simon

# Informations générales sur le document

### Contact

Pour toute question ou remarque concernant ce document, merci de contacter :

Wattin Jérôme - Etudiant

Téléphone : +32479886237

Mail : jerome.wattin@std.heh.be

HEH Campus Technique

8A Avenue Maistriau,

7000 Mons

### Confidentialité

Ce document contient des informations confidentielles et exclusives de la Haute École en Hainaut (HEH). Le service informatique ne peut divulguer les informations confidentielles contenues dans ce document à un tiers sans le consentement écrit de la HEH, hormis aux employés, enseignants ou directeurs qui ont besoin de connaître son contenu à des fins d'évaluation du document. Le service informatique se doit d'informer ces personnes de la nature confidentielle de ce document et d'obtenir leur accord pour préserver sa confidentialité.

### Termes et conditions

La HEH n'assume aucune responsabilité pour les erreurs ou omissions dans le contenu de ce document ou de tout document de tiers référencé ou associé, y compris, mais sans s'y limiter, les erreurs typographiques, les inexactitudes ou les informations périmées. Ce document et tous les renseignements qui s'y trouvent sont fournis «tels quels» sans aucune garantie, expresse ou implicite.

### Informations sur le document

|  |  |
| --- | --- |
| Nom du document | RobotBaliseurQ2.Docx |
| Version | Version 1.00 |
| Niveau de confidentialité | Utilisation interne uniquement |
| Auteur du document | Wattin Jérôme |
| Contributeur(s) |  |
| Révisé par |  |
| Approuvé par |  |

### Versions

|  |  |  |  |
| --- | --- | --- | --- |
| version | Date de parution | Modification réalisée par | Modification(s) apportée(s) |
| 1.00 | 26/03/2018 | Wattin.J | Création du document |
|  |  |  |  |
|  |  |  |  |

### Documents connexes et/ou de référence

|  |  |  |  |
| --- | --- | --- | --- |
| Nom du document | Description | Date | version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Table des matières

[1. Informations générales sur le document 3](#_Toc514867043)

[Contact 3](#_Toc514867044)

[Confidentialité 3](#_Toc514867045)

[Termes et conditions 3](#_Toc514867046)

[Informations sur le document 4](#_Toc514867047)

[Versions 4](#_Toc514867048)

[Documents connexes et/ou de référence 4](#_Toc514867049)

[Table des matières 5](#_Toc514867050)

[2. Introduction et présentation générale du projet 6](#_Toc514867051)

[3. Matériel utilisé 6](#_Toc514867052)

[4. Diagramme de Gantt 8](#_Toc514867053)

[5. Présentation détaillée du capteur à ultrason 8](#_Toc514867054)

[5.1. Caractéristiques 8](#_Toc514867055)

[5.2. Brochage 9](#_Toc514867056)

[5.3. Physique et fonctionnement 9](#_Toc514867057)

[5.4. Programmation 10](#_Toc514867058)

[6. Contrôle du robot 11](#_Toc514867059)

[7. Choix de l’IDE 13](#_Toc514867060)

[8. Outils utilisés pour la gestion du projet 13](#_Toc514867061)

[9. Algorithme de programmation 15](#_Toc514867062)

[9.1. Programmation du robot 15](#_Toc514867063)

[10. Problèmes rencontrés 16](#_Toc514867064)

[11. Améliorations 17](#_Toc514867065)

[12. Conclusions 17](#_Toc514867066)

[13. Annexe 18](#_Toc514867067)

# Introduction et présentation générale du projet

Ce rapport présente la réalisation et le résultat d’un projet, réalisé sur une semaine, ayant pour but de programmer une voiture, en lui assignant une tâche spécifique.

Nous avions plusieurs choix pour celle-ci, mais nous nous sommes portés sur un robot baliseur. Ce projet étant intéressant sur plusieurs points : il implique des calculs physiques (capteurs, calcul de distances), ainsi qu’un algorithme de déplacement autonome.

Celui-ci devait, de manière indépendante, de tracer les contours d’une pièce, et de calculer les dimensions de celle-ci.

Lors de la répartition des tâches, et après réflexion, nous avons décidé d’implémenter une fonction supplémentaire à celui-ci. Le contrôle et l’affichage du dit tracé, sur un site web hébergé sur le Raspberry Pi.

Afin de réaliser ces différentes fonctions, nous avons utilisé plusieurs langages :

* Python pour l’algorithme de déplacement ainsi que le tracé du déplacement.
* PHP, HTML, MySQL et CSS pour l’interface du site internet.

# Matériel utilisé

Pour réaliser cela, nous avons utilisé un châssis de voiture 4tronix Initio, composée de 4 roues et 2 moteurs pas-à-pas.

Chaque moteur contrôle 2 roues simultanément, et du même côté. Cela permet une rotation pratiquement sur place.

Pour alimenter le robot, un roulement de 2 batteries externes a été effectué, afin d’éviter une sous-alimentation. Lors de l’utilisation, la tension de sortie baissait, ce qui avait tendance à impacter le fonctionnement des capteurs et moteurs.

Nous avons également utilisé 2 capteurs à ultrason HC-SR04 afin de détecter les obstacles ainsi qu’un Raspberry pi 3B pour programmer le robot, et récupérer les données des capteurs.

Afin de sécuriser, et améliorer son déplacement, nous avons fixé un capteur à ultrason à l’avant du robot et l’autre sur le côté gauche.

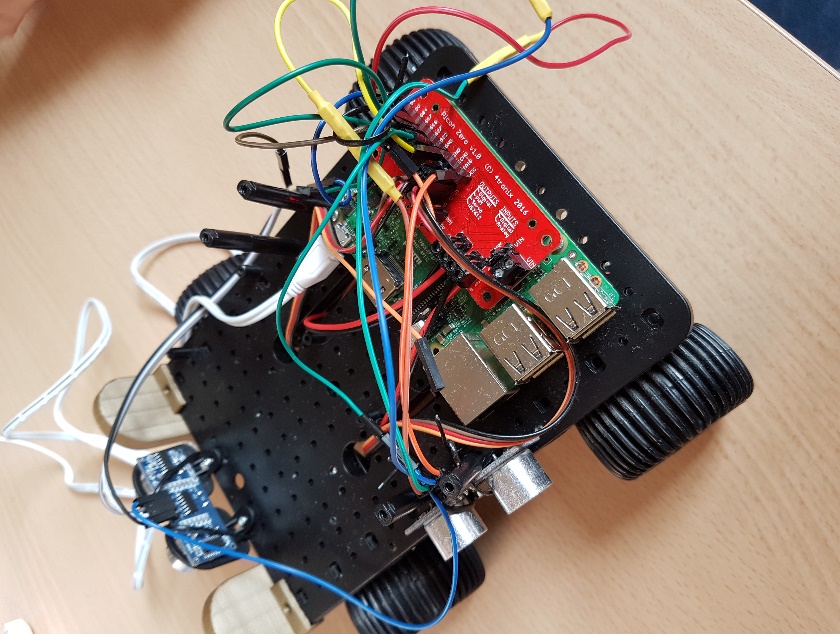
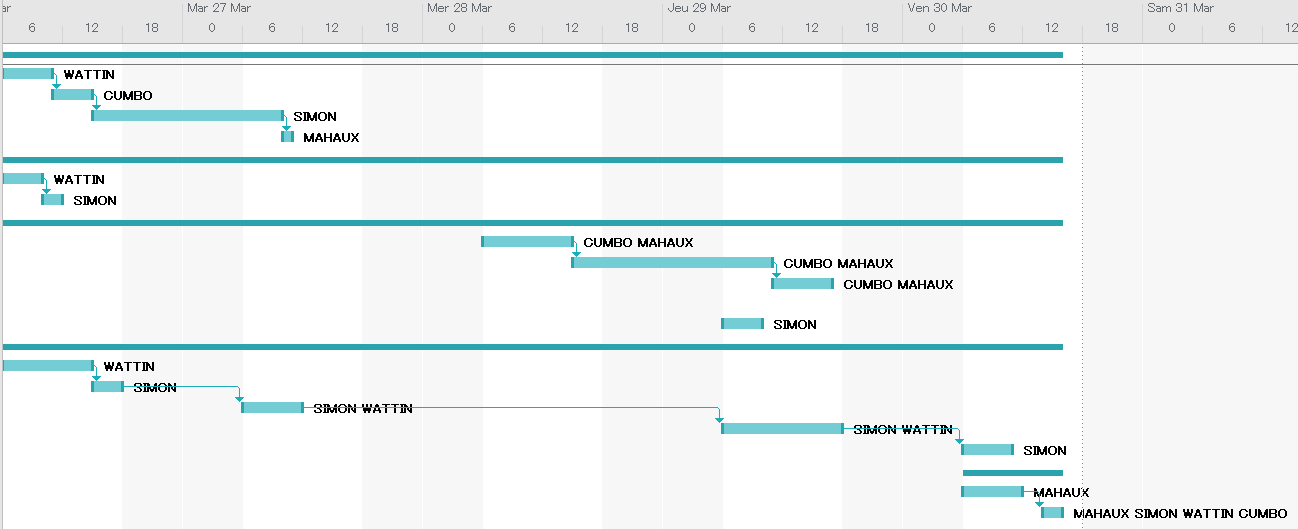


Photo du robot monté



Photo du robot monté

# Diagramme de Gantt



# Présentation détaillée du capteur à ultrason

Le capteur à ultrason fonctionne sur base de la vitesse du son, afin de déterminer la distance le séparant d’un obstacle. Il fonctionne sans tenir compte des variables de lumière (la lumière du soleil, la couleur de l’obstacle).

## Caractéristiques

Dimensions : 45 mm x 20 mm x 15 mm

Plage de mesure : 2 cm à 400 cm

Résolution de la mesure : 0.3 cm

Angle de mesure efficace : 15 °

Largeur d'impulsion sur l'entrée de déclenchement : 10 µs (broche Trigger)

## Brochage



Vcc = Alimentation +5V DC

Trig = Entrée de déclenchement de la mesure (Trigger input)

Echo = Sortie de mesure des données en écho (Echo output)

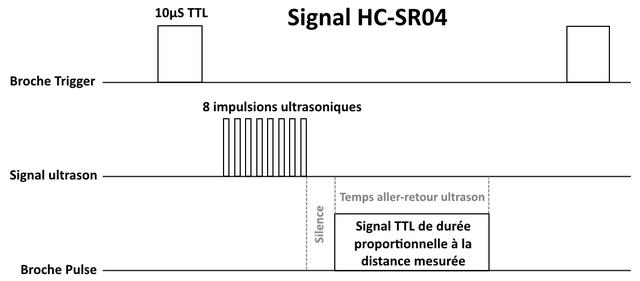
GND = Masse

## Physique et fonctionnement

La prise d’une mesure se déroule comme suit :

1. On envoie une impulsion HIGH durant 10 µs sur la broche Trigger (pin nommé « Trig ») du capteur
2. Le capteur répond à l’impulsion, par l’envoie d’une série de 8 impulsions ultrasonique (40KHz), inaudible pour l’être humain.
3. À l’impact d’un objet, l’ultrason retourne en sens inverse. Cela signifie que lors d’un impact face à la source, l’onde est renvoyée en direction du capteur
4. lors de la réception de l’ECHO, le capteur clôture.

Afin de déterminer la durée de l’aller-retour de l’onde, la broche ECHO du capteur (pin nommé « echo ») reste sur HIGH durant les étapes 3 et 4.



Dans le manuel du capteur, nous avons trouvé la formule d=durée/58 cm.

Celle-ci n’ayant aucune explication, et n’étant peu explicite, nous allons la calculer.

Nous pouvons calculer une vitesse avec la formule *d=v\*t* (distance = vitesse \* temps).

De ce fait, on peut calculer la distance parcourue par un son, en multipliant la vitesse du son (à peu près 340m/s) par le temps de propagation.

Le capteur envoie une impulsion d’une dure de 10 µs. Sachant que la valeur retournée est celle d’un aller-retour, nous devons donc multiplier 10µs par la valeur, puis diviser par 2.

Sachant que v = 340m/s, soit 34 000cm/ 1 000 000µs.

*d = 34 000cm /1 000 000µs \* 10µs \* valeur/2*

*d = 170 0000/1 000 000 cm \* valeur*

*d = 17/100cm \* valeur*

Cette formule nous permet, simplement avec la valeur que retourne le capteur, de déterminer la distance le séparant de l’obstacle.

## Programmation

La fonction permettant de récupérer, et renvoyer la distance fonctionne comme suit :

Définition de la vitesse du son en mètre par seconde :

mtrs\_per\_sec = 343

Envoie d’une impulsion de 10µs sur le pin trigger :

GPIO.output(self.\_trigger\_pin, True)

time.sleep(0.00001)

GPIO.output(self.\_trigger\_pin, False)

Ecoute du pin echo pour le return :

while GPIO.input(self.\_echo\_pin) == 0:

echoStart = time.time()

break

Quand le pin reçoit la réponse :

while GPIO.input(self.\_echo\_pin) == 1:

echoStop = time.time()

Ce qui permet de calculer la durée entre l’envoi et la réception de l’impulsion :

echoTime = echoStop – echoStart

Enfin, cela permet de calculer la distance (en centimètres) :

distance = (echoTime \* self.\_mtrs\_per\_sec \* 100) / 2

# Contrôle du robot

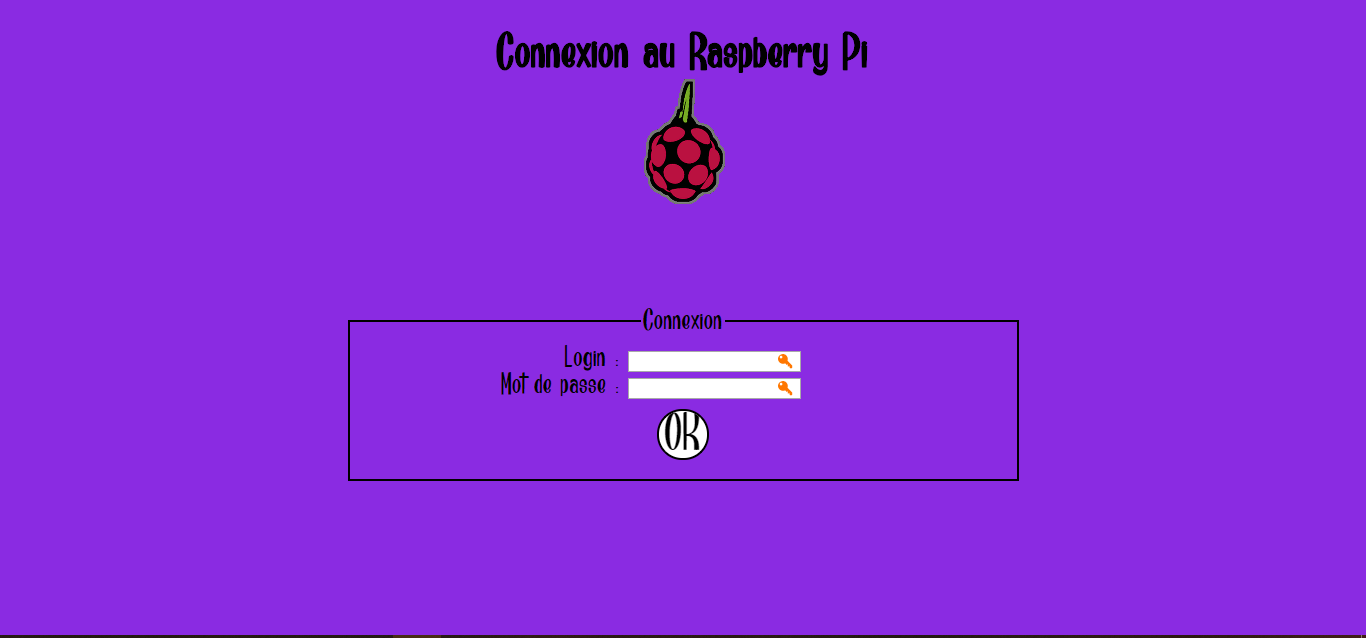
Pour d’exploiter au mieux les capacités du Raspberry pi 3, ainsi que pour le côté innovation du projet, nous avons décidé de développer un site web. Cette solution nous a permis à fournir un contrôle du robot compatible avec pratiquement tous les supports.

L’objectif premier du site était d’obtenir une interface permettant de démarrer le robot, de récupérer son tracé, et l’afficher dynamiquement.

Nous avons donc dû installer un serveur Apache 2 sur le Raspberry pi 3 afin de répondre aux requêtes envoyées par l’utilisateur.

Nous avons également installé MySQL et PHP sur celui-ci.

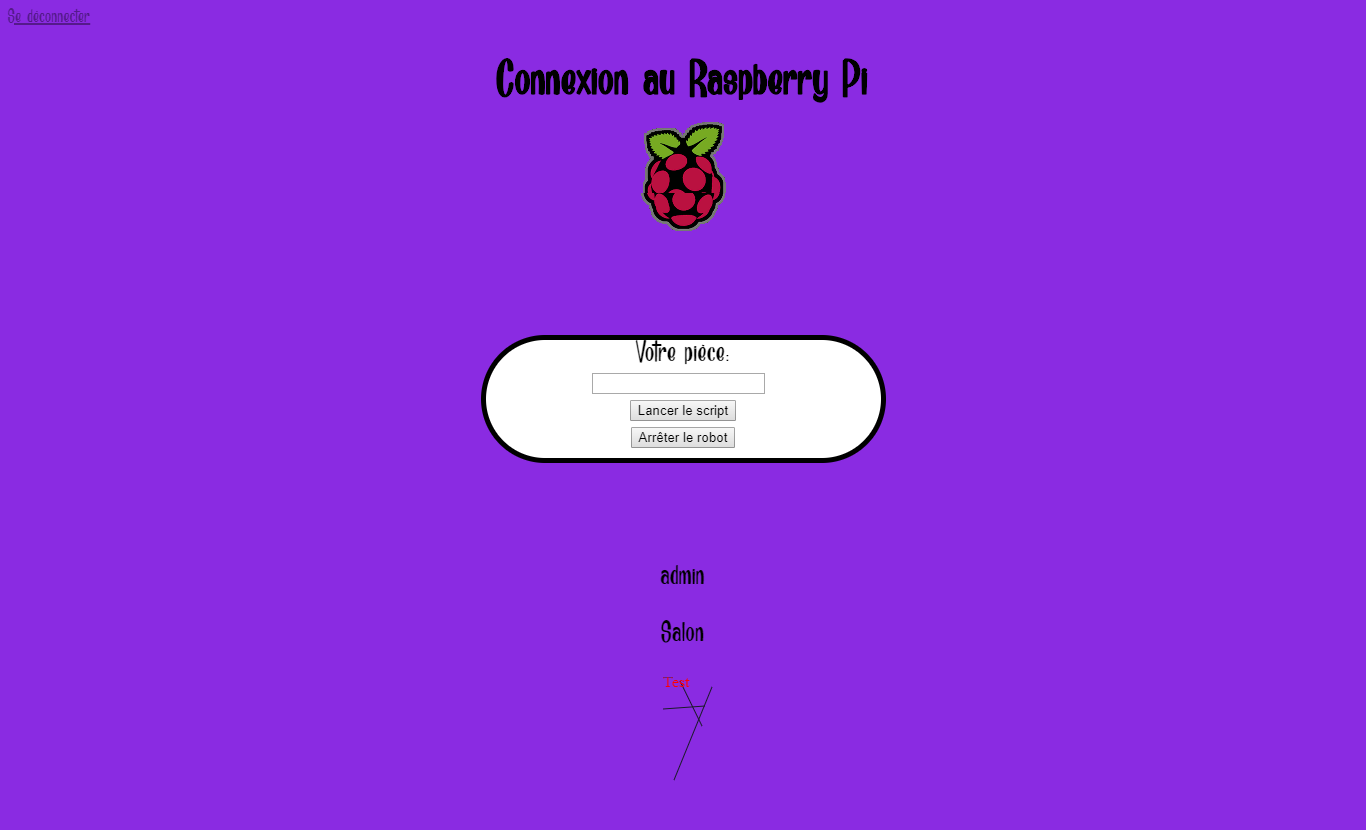
Un minimum de sécurité sur le site web a été implémenté afin d’éviter que n’importe qui puisse démarrer une cartographie. Une interface de connexion a donc été mise en œuvre. Pour stocker les comptes utilisateurs et se connecter, nous avons ajouté une table appropriée à notre base de données.



Interface de connexion du site web

Afin d’apporter un intérêt supplémentaire au site web, nous avons également créé une autre table dans la base de données pour stocker l’utilisateur, la pièce indiquée et sa cartographie.

Chaque utilisateur peut donc visualiser son historique de cartographie effectué.



Interface permettant de lancer le script pour cartographier et de visualiser ses cartographies déjà effectuées

# Choix de l’IDE

Notre choix d’IDE s’est porté sur Geany.  
En effet, celui-ci étant léger et équipé d’un interpréteur intégré, il inclut les fonctionnalités élémentaires pour la réalisation de notre projet. De plus il est multi-plateforme et supporte les langages que nous avions dû utiliser, c’est-à-dire Python, HTML, CSS et PHP.

Il répondait donc à nos exigences.

# Outils utilisés pour la gestion du projet

Afin de permettre une gestion de projet efficace, nous avons tous utilisé Git et créé un dépôt sur Github.

Cela nous a principalement permis de synchroniser toutes nos modifications effectuées. Tout le monde a donc pu travailler en parallèle sur le projet ce qui a amélioré la gestion générale de celui-ci et nous avons également pu gérer les différentes versions du robot baliseur.

L’utilisation de Github pourra nous permettre en autre de partager notre projet à une plus large communauté.

Nous avons également utilisé Microsoft Project pour visualiser la répartition des tâches et obtenir le Diagramme de Gantt.

# Algorithme de programmation

## Programmation du robot

Paroi = false

Tant que TRUE :

devant = Capture donnée capteur devant

gauche = Capture donnée capteur gauche

TANT QUE devant > 1000 ou gauche > 1000 : // boucle en cas de valeurs erronées du capteur

Sample = 3, prise de données, sample = 1 //on définit sample à 3 pour donnée + précise

If paroi = true : // Si le robot a touché la première paroi

Si capteurDevant < 50 et capteurDevant!= 0 : // Détection d’un mur devant le robot

Tant que 30 > capteurDevant > 10 // tâtonnement pour se rapprocher au max du mur

Avance un peu, puis s’arrête

Rotation 90° à droite // proche du mur, rotation à droite

Autre si devant > 30 et ( gauche > 50 ou gauche == 0) : //détection d’un tournant gauche

Rotation 90° à gauche

Autre si devant >75 et ( gauche < 45 et gauche != 0) : // détection d’une longue paroi à longer

Sample = 4, prise de données, tableau side[] = donnée gauche //stock de la distance à gauche dans un tableau

si side.longueur > 2 : //si tableau comporte 2 données minimum

si side[-1] < side[-2] //comparaison des 2 dernières données stockées

rotation gauche légère en fonction de la différence//le robot penche vers l’extérieur

sinon : rotation droite légère en fonction de la différence//le robot penche vers l’intérieur

else : // si le robot n’a pas encore touché la première paroi

robot avance vitesse maximale

Si capteurDevant < 50 et capteurDevant!= 0 : // Détection d’un mur devant le robot

Tant que 30 > capteurDevant > 10 // tâtonnement pour se rapprocher au max du mur

Pz.forward (60), pause(0.2sec), pz.stop() // avance un peu, puis s’arrête

If devant < 10 : // proche du mur, rotation à droite

Rotation 90° à droite

Paroi = true

# Problèmes rencontrés

* Les premiers problèmes auxquels nous avons été confrontés ont été la fixation et le montage des 2 capteurs à ultrason sur le PiconZero.

En effet, étant donné que ce dernier ne possédait qu’une seule entrée TRIG et ECHO, nous avons dû nous rabattre sur les ports GPIO du Raspberry Pi 3 pour brancher et faire fonctionner les 2 capteurs à ultrason.

* L’exécution de script Python par le site WEB et donc par le code PHP nous a particulièrement posé problème.  
  En effet, par défaut l’utilisateur créé par Apache (www-data) n’a pas les droits pour réaliser certaines actions, il a donc fallu éditer le fichier sudoers via la commande *sudo visudo* et donner les droits à l’utilisateur www-data sur les scripts visant à être exécutés.   
  Il a également fallu mettre l’utilisateur www-data comme propriétaire avec le groupe root sur les fichiers PHP et Python voulu.
* Nous avons eu des soucis pour permettre d’arrêter le script Python directement depuis le site web.   
  Effectivement étant donné que lorsque l’on démarrait une cartographie, un script Python pour cartographier était appelé et tant que son exécution n’était pas terminée, la page web indiquée toujours un chargement puisque le code PHP n’avait également pas fini d’être exécuté.   
  Dès lors lorsque l’utilisateur souhaité stopper la cartographie, le code PHP permettant d’appeler un autre script Python pour arrêter le robot ne pouvait donc pas être exécuté tant que le navigateur chargé toujours la page.  
  Pour arrêter le robot depuis le site web il fallait donc cliquer sur la croix du navigateur et ensuite cliquer sur le bouton pour arrêter la cartographie.
* Après de nombreux essais, nous nous sommes rendu compte que lorsque la batterie alimentant notre robot n'était pas entièrement chargée, le capteur pouvait retourner des valeurs erronées. Nous avons donc dédié une boucle dans notre algorithme, excluant ces valeurs, afin d'éviter tous soucis.
* Lors d'un déplacement en ligne droite, le robot avait une fâcheuse tendance à dévier, en fonction de la légère inclinaison du sol, ainsi que des différences de revêtement.

Une fonction de correction de trajectoire a été développée, afin de corriger dynamiquement la trajectoire de celui, et le garder constamment à une distance comprise entre 20 et 30 cm du mur.

* Encore une fois, suite à des différences du revêtement de sol (et donc adhérence du robot), les déplacements en angle n'étaient pas toujours de même amplitude. Nous n'avons malheureusement pas trouvé de solution fiable à ce problème.

# Améliorations

* Au niveau du site Web nous aurions pu améliorer la fonction permettant d’arrêter le robot depuis le site web avec l’utilisation de JavaScript et d’AJAX.

Nous aurions pu lancer le script python en AJAX (XMLHttpRequest) et l’arrêter avec du JavaScript (req.abort()) pour résoudre ce problème, mais nous n’avons pas eu le temps pour effectuer cela.

* Nous pourrions également améliorer le front-end et le back-end du site.   
  En effet il serait plus convivial pour l’utilisateur d’afficher les différentes cartographies effectuées dans un tableau et avec une taille plus grande.

Au niveau du back-end, il serait intéressant d’améliorer l’ensemble du code PHP pour éviter les répétitions en adoptant par exemple le design pattern MVC, mais étant limité au niveau de temps nous n’avons pas pu réaliser cela, nous nous sommes davantage concentré sur le bon fonctionnement de la cartographie.

* L’algorithme du robot ne comporte pas de fonction, cependant nous avons perdu énormément de temps à compenser les différentes constantes physiques (adhérence du sol, inclinaison, baisse de tension de la batterie). Il aurait été optimal de créer des fonctions pour chaque action à réaliser pour le robot pour obtenir un code plus ordonné.

# Conclusions

En conclusion, nous pouvons dire que ce projet était intéressant à réaliser et nous a apporté de nouvelles notions.

En effet, ce travail nous a appris à mieux gérer la gestion générale d’un projet en répartissant efficacement le temps et le travail à fournir grâce à des stratégies de répartition des tâches. De plus l’utilisation d’outils en ligne pour synchroniser nos travaux comme GitHub a davantage contribué à cet apprentissage.

Ce projet nous a également permis d’en apprendre plus sur le fonctionnement du Raspberry Pi 3 et de l’intérêt d’en utiliser un pour ce genre des projets mêlant l’informatique à l’électronique.

En outre, avons pu étudier le fonctionnement d’un capteur à ultrason, notamment les principes physiques et leur application en programmation.

Nous avons également développé nos compétences sur les langages utilisés lors du projet. Particulièrement sur Python et PHP grâce au travail de recherche réalisé, ce qui devrait nous être utile pour la suite de notre cursus.

Ce projet s’est donc dans l’ensemble bien déroulé pour toutes ces raisons mentionnées, mais aussi grâce à une bonne entente au sein du groupe.

# Annexe

Tous les codes