

**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](#_Toc514846574)

[Contact 3](#_Toc514846575)

[Confidentialité 3](#_Toc514846576)

[Termes et conditions 3](#_Toc514846577)

[Informations sur le document 4](#_Toc514846578)

[Versions 4](#_Toc514846579)

[Documents connexes et/ou de référence 4](#_Toc514846580)

[Table des matières 5](#_Toc514846581)

[2. Introduction et présentation générale du projet 6](#_Toc514846582)

[3. Matériel utilisé 6](#_Toc514846583)

[4. Diagramme de Gantt 7](#_Toc514846584)

[5. Présentation détaillée du capteur à ultrason 7](#_Toc514846585)

[6. Contrôle du robot 8](#_Toc514846586)

[7. Choix de l’IDE 10](#_Toc514846587)

[8. Outils utilisés pour la gestion du projet 10](#_Toc514846588)

[9. Algorithme de programmation 11](#_Toc514846589)

[9.1. Programmation du robot 11](#_Toc514846590)

[9.2. Développement du site web 15](#_Toc514846591)

[10. Problèmes rencontrés 15](#_Toc514846592)

[11. Améliorations 16](#_Toc514846593)

[12. Conclusions 16](#_Toc514846594)

[13. Annexe 16](#_Toc514846595)

# Introduction et présentation générale du projet

L’objectif principal de ce projet a été de programmer une voiture afin de lui permettre de réaliser une cartographie autonome de son environnement.

Nous avons été poussé à choisir le robot baliseur pour le côté original du projet et parce que nous avons estimé le sujet intéressant ainsi que le travail de recherche derrière qui pourraient nous apporter de nouvelles notions.

# Matériel utilisé

Pour réaliser cela, nous avons utilisé une base d’un robot de marque Initio composée d’un châssis, 4 roues et 2 moteurs pas à pas.

Nous avons également utilisé 2 capteurs à ultrason pour détecter les obstacles ainsi qu’un Raspberry pi 3B pour programmer le robot.

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

Enfin, nous avons utilisé une batterie 5V 2A pour alimenter tous les composants.

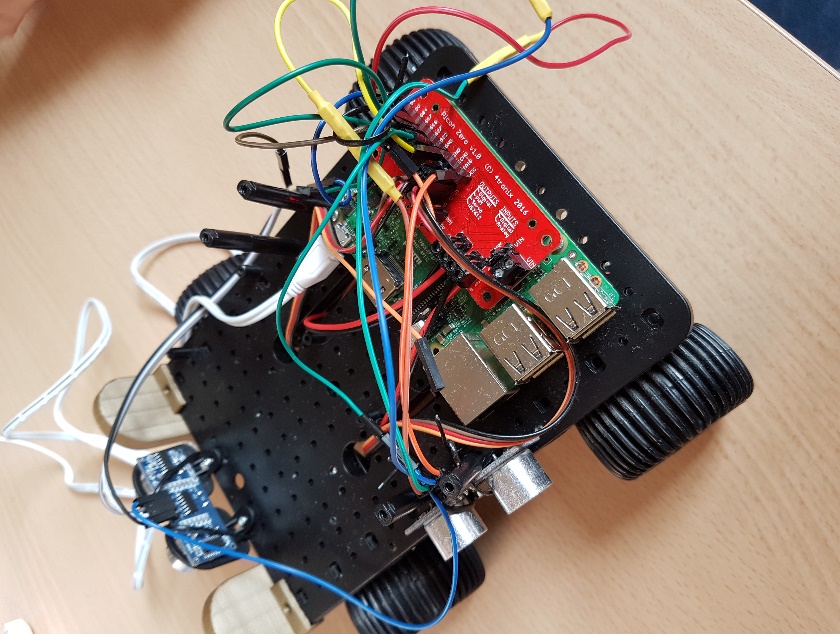


Photo du robot monté



Photo du robot monté

# Diagramme de Gantt

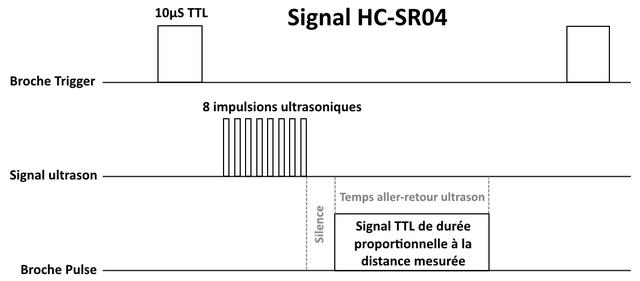
# Présentation détaillée du capteur à ultrason

**PRINCIPE PHYSIQUE, ELECTRONIQUE ET PROGRAMMATION DEMANDEE, PARTIE MICHIELS DONC IMPORTANT**

[**https://www.gotronic.fr/pj2-hc-sr04-utilisation-avec-picaxe-1343.pdf**](https://www.gotronic.fr/pj2-hc-sr04-utilisation-avec-picaxe-1343.pdf)

**https://www.carnetdumaker.net/articles/mesurer-une-distance-avec-un-capteur-ultrason-hc-sr04-et-une-carte-arduino-genuino/**

Le capteur à ultrason fonctionne sur base de la vitesse du son.



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. A 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’écho, 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.

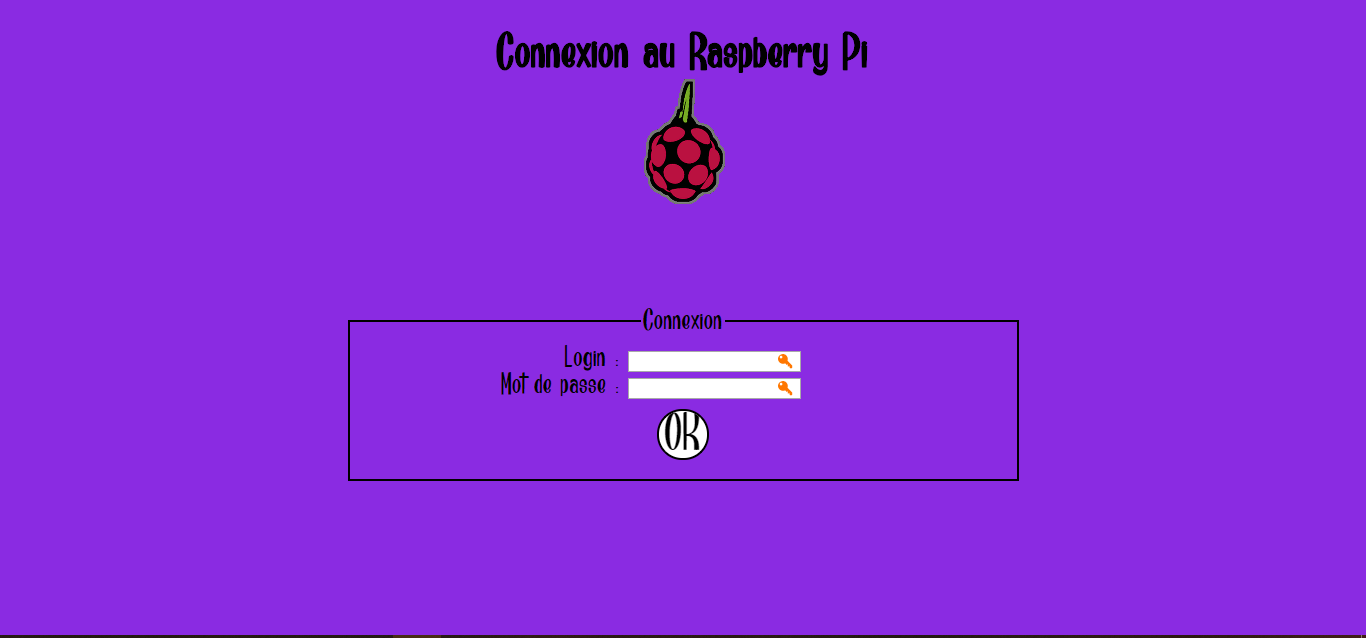
# Contrôle du robot

Pour tirer parti au mieux du Raspberry pi 3 et pour le côté « innovation » du projet, nous avons décidé de développer un site web. Cette solution nous a permis à ce que le contrôle du robot soit opérationnel avec pratiquement tout les supports.

L’objectif premier du site était d’obtenir une interface pour démarrer le robot afin de cartographier une pièce et l’afficher en retour.

Nous avons donc dû installer un serveur Apache 2 sur le Raspberry pi 3 permettant de répondre aux requêtes envoyées par l’utilisateur, et nous avons également installé MySQL et PHP sur celui-ci.

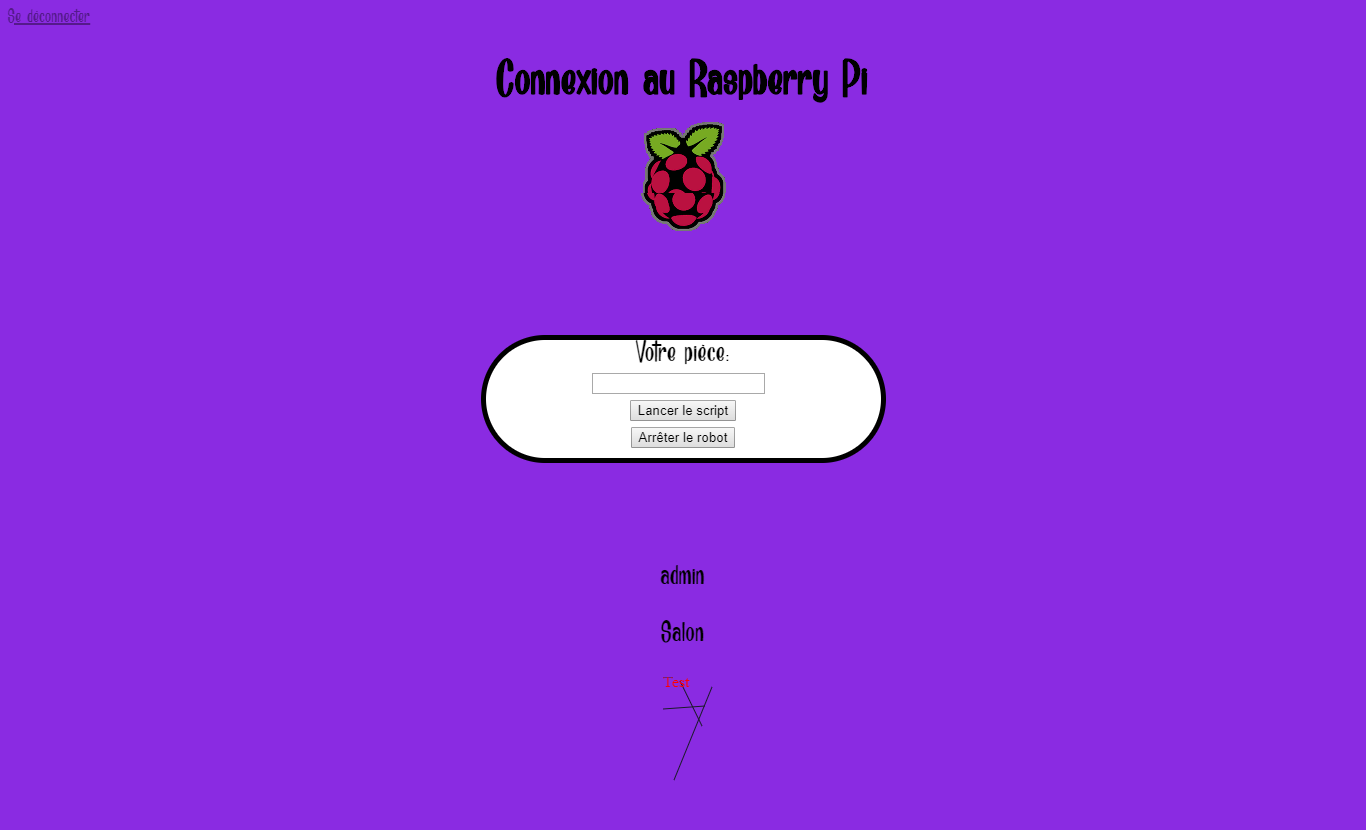
Un minimum de sécurité sur le site web a été implémenté afin d’éviter que n’importe qui puisse démarrer une cartographie. Une interface de connexion a donc été mise en œuvre. Pour stocker les comptes utilisateurs et se connecter, nous avons ajouté une table appropriée à notre base de données.



Interface de connexion du site web

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

Chaque utilisateur peut donc visualiser son historique de cartographie effectué en fonction de sa pièce.



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 nous avons aussi créée 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

*Mapper.py*

1. *#!/bin/python*
3. *# Import des librairies*
4. **import** piconzero **as** pz
5. **import** time
6. **from** Bluetin\_Echo **import** Echo
7. **from** threading **import** Thread
9. *# Definition des pins*
10. TRIGGER\_PIN1 = 27
11. ECHO\_PIN1 = 22
12. TRIGGER\_PIN2 = 18
13. ECHO\_PIN2 = 17
15. *# Definition de la constante physique vitesse du son pour permettre le calcul de la distance entre capteur/obstacle*
16. speed\_of\_sound = 315
18. *# Definition du nombres d echantillon a chaque prise de donnee du capteur (moyenne si plusieur)*
19. samples = 1
21. *# Definition des distances min et max entre robot et mur pour l'algorithme de redressement de trajectoire*
22. gapmin = 20
23. gapmax = 30
25. *# Vitesse du robot*
26. speed = 100
28. *# Le robot ne suis pas encore un mur au depart*
29. onTrack = False
31. pz.init( )
32. **while** True:
34. *# prise de donnees du capteur avant*
35. capteurDevant = Echo(TRIGGER\_PIN1, ECHO\_PIN1, speed\_of\_sound)
36. front = capteurDevant.read('cm',samples)
37. capteurDevant.stop()
39. *# prise de donnees du capteur gauche*
40. capteurGauche = Echo(TRIGGER\_PIN2, ECHO\_PIN2, speed\_of\_sound)
41. side = capteurGauche.read('cm',samples)
42. capteurGauche.stop()
43. **print**("devant = ",front," gauche = ",side)
45. *# boucle de correction quand le capteur renvoie des valeurs incoherentes*
46. **while** front > 1000 **or** side > 1000:
47. **print**("Erreur du capteur, nouvelle prise de donnees")
48. *# arret du robot*
49. pz.stop()
50. *# definition de 3 echantillons pour capture plus precise*
51. samples = 3
52. capteurDevant = Echo(TRIGGER\_PIN1, ECHO\_PIN1, speed\_of\_sound)
53. front = capteurDevant.read('cm',samples)
54. capteurDevant.stop()
56. capteurGauche = Echo(TRIGGER\_PIN2, ECHO\_PIN2, speed\_of\_sound)
57. side = capteurGauche.read('cm',samples)
58. capteurGauche.stop()
59. **print**("devant = ",front," gauche = ",side)
60. *#remise a 1 de l'echantillon*
61. samples = 1
62. paroi = False
63. *#boucle une fois avoir rejoint une paroi*
64. **if** onTrack :
66. *# Detection du mur a proximite*
67. **if** front < 50 **and** front != 0 :
68. pz.forward(40)
69. **while** front < gapmax :
70. **print**(front,"Je detecte une paroi en face")
71. *# tatonnement pour se rapprocher au maximum du mur*
72. **while** gapmax > front > 10 :
73. **print**("je peux me rapprocher distance = ",front)
74. pz.forward(60)
75. time.sleep(0.2)
76. pz.stop()
77. capteurDevant = Echo(TRIGGER\_PIN1, ECHO\_PIN1, speed\_of\_sound)
78. front = capteurDevant.read('cm',samples)
79. capteurDevant.stop()
80. *#proximite max du mur, sequence de rotation a 90 droite*
81. **if** front < 10 :
82. pz.stop()
83. pz.forward(-50)
84. time.sleep(0.5)
85. pz.spinRight(90)
86. time.sleep(1)
87. pz.forward(50)
88. capteurDevant = Echo(TRIGGER\_PIN1, ECHO\_PIN1, speed\_of\_sound)
89. front = capteurDevant.read('cm',samples)
90. capteurDevant.stop()
92. *# Detection d un angle de 90 a gauche*
93. **elif** front > gapmax **and** ( side > 50 **or** side == 0):
94. **print**("je detecte un tournant a gauche, je tourne a gauche")
95. pz.forward(-65)
96. time.sleep(0.2)
97. pz.spinLeft(90)
98. time.sleep(1.1)
99. pz.forward(100)
100. time.sleep(0.5)
102. *# Algorithme de correction de trajectoire quand longue distance devant et a proximite d un mur*
103. **elif** front > 75 **and** (side < gapmax **and** side != 0):
104. **print**("Je longe un mur")
105. deport = [] *#creation tableau et stock des donnes laterales*
107. *#boucle tant que le robot a de l espace devant lui et pas de virage gauche detecte*
108. **while** front > 75 **and** (side < gapmax+15 **and** side != 0):
109. *# pour un meilleur fonctionnement, 4 echantillons par prise*
110. sample = 4
111. capteurDevant = Echo(TRIGGER\_PIN1, ECHO\_PIN1, speed\_of\_sound) *#capture des distances de devant et gauche*
112. front = capteurDevant.read('cm',samples)
113. capteurDevant.stop()
114. capteurGauche = Echo(TRIGGER\_PIN2, ECHO\_PIN2, speed\_of\_sound)
115. side = capteurGauche.read('cm',samples)
116. capteurGauche.stop()
117. **print**("fction devant = ",front," gauche = ",side)
118. *#si la valeur gauche est differente de 0 (bug occassionel), ajout de la valeur dans le tableau*
119. **if** side != 0 : deport.append(side)
120. compar = len(deport) *#attribution de la longueur du tableau a compar*
121. **print**("longueur du tableau",compar)
122. *#dans le cas ou le robot est beaucoup trop proche, fonction dediee a le remettre a une distance correcte*
123. **if** side < 10 **and** side != 0 :
124. **print**("Je suis trop proche",side)
125. **if** (20-side)/10 > 0.1 :
126. pz.spinRight(65)
127. time.sleep((20-side)/10)
128. pz.forward(60)
129. time.sleep(0.2)
130. pz.spinLeft(65)
131. time.sleep((20-side)/10)
132. pz.forward(50)
133. *#dans le cas ou le robot est beaucoup trop loin, fonction dediee a le rapprocher*
134. **elif** 40> side > 20 **and** side != 0 :
135. **print**("je suis trop loin",side)
136. **if** (side-20)/10 > 0.1 :
137. pz.spinLeft(65)
138. time.sleep((side-20)/10)
139. pz.forward(60)
140. time.sleep(0.4)
141. pz.spinRight(50)
142. time.sleep((side-20)/10)
143. pz.forward(50)
145. **if** compar >= 2: *#si le tableau comporte au moins 2 donnees, on peut commencer la correction de trajectoire*
146. **print**("deport-1 = ",deport[-1], "deport -2 =", deport[-2])
147. **if** deport[-1] > deport[-2]: *#on determine si le robot devie interieurement ou exterieurement*
148. **print**("deport exterieur")
149. deportInterieur = False *#*
150. **else** :
151. deportInterieur = True
152. **print**("deport interieur")
153. **if** deportInterieur == True : *# on calcule le coefficient de deport et on effectue un angle en fonction*
154. coef = (deport[-2]-deport[-1])/10
155. **print**("coef = ",coef)
156. pz.spinRight(50)
157. time.sleep(coef)
158. pz.forward(50)
159. **elif** deportInterieur == False :
160. coef = (deport[-1]-deport[-2])/10
161. **print**("coef = ",coef)
162. pz.spinLeft(50)
163. time.sleep(coef)
164. pz.forward(50)
166. **else** :
167. pz.forward(speed)
169. **else** : *#fonction de depart, pour que le robot aille droit tant qu il ne rencontre pas de mur*
170. **print**("Debut de la sequence")
172. *# Toujours tout droit tant qu'on approche pas d'un obstacle*
173. **if** front > gapmax **and** (side > gapmax **or** side == 0):
174. **if** front < 50:
175. speed = 40
176. **print**("Mur a proximite, reduction de la vitesse")
177. pz.forward(speed)
178. *# Quand on croise un obstacle on suit une piste*
179. **else** :
180. pz.stop()
181. "Premier mur detecte, debut de la sequence detection"
182. onTrack = True
184. time.sleep(0.1)

## Développement du site web

# 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 du nous rabattre sur les port GPIO du Raspberry Pi 3 pour brancher et faire fonctionner les 2 capteurs à ultrason.  
Au niveau des fixations des capteurs, nous avons dû réutiliser celles du toit afin de pouvoir les poser latéralement. ??

* 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 créée 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 démarré une cartographie, un script Python pour cartographier était appelé et tant que son exécution n’était pas terminée, la page web indiqué toujours un chargement puisque le code PHP n’avait également pas finis 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.   
  Effectivement, 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 donc 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.
* SVG…

# 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.

* SVG

# Conclusions

# Annexe

Tous les codes