HAUTE ECOLE EN HAINAUT
Campus technique
8a, avenue V. Maistriau B-7000 Mons
Tél : +32 (0)65 33 81 54
Fax : +32 (0)65 31 30 51
E-mail : tech-mons@heh.be
www.heh.be

Nano Ordinateurs
Gestion de projet
Robot Baliseur

Mai 2018
Version 3.0

Wattin Jérôme
Sam Mahaux
Fabio Cumbo
Timothée Simon

Formations de Bacheliers et de Masters organisées par la Fédération Wallonie-Bruxelles
FÉDÉRATION WALLONIE-BRUXELLES
pôle hainuyer

1. 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 3.00
Niveau de confidentialité	Utilisation interne uniquement
Auteur du document	Wattin Jérôme, Cumbo Fabio, Simon Timothée, Mahaux Sam
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
2.00	20/05/2018	Wattin.J, Cumbo.F, Simon.T, Mahaux.S	Ajout des parties du projet de chaque membre du groupe
3.0	24/05/2018	Wattin.J, Cumbo.F, Simon.T, Mahaux.S	Correction de l'orthographe et ajout des codes en annexe

Documents connexes et/ou de référence

Nom du document	Description	Date	version

Table des matières

2. 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, permettre 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.

3. 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.

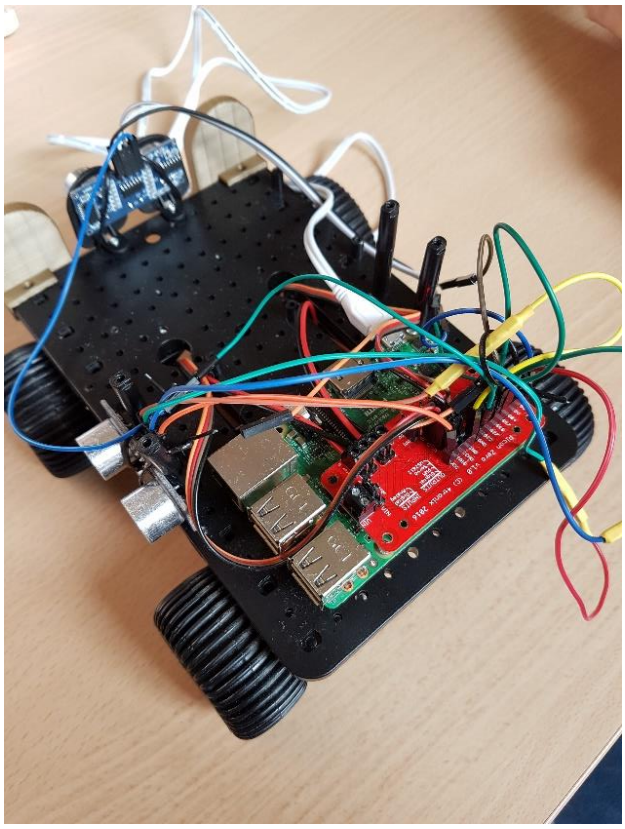
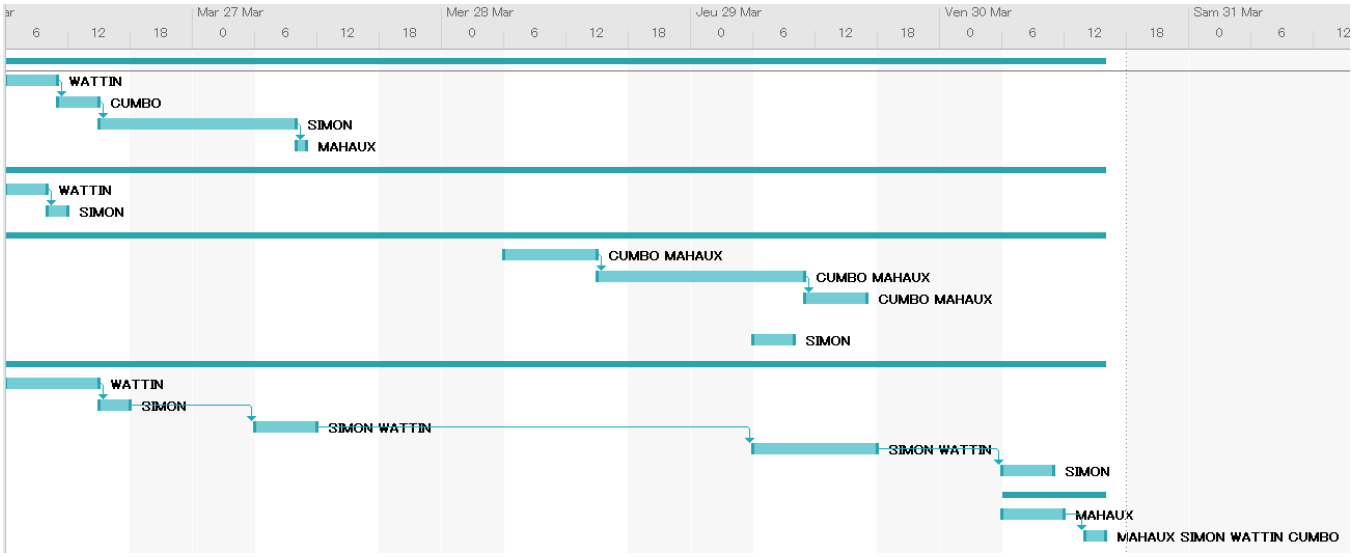


Photo du robot monté 1

Photo du robot monté 2

4. Diagramme de Gantt



5. 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).

5.1. 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)

5.2. Brochage

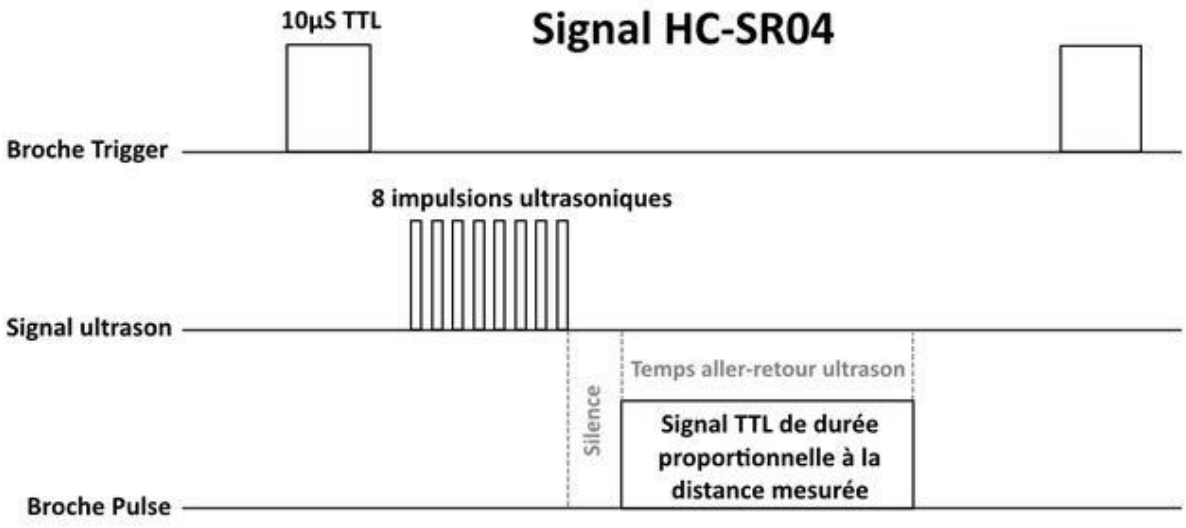
Vcc = Alimentation +5V DC
Trig = Entrée de déclenchement de la mesure (Trigger input)
Echo = Sortie de mesure des données en écho (Echo output)
GND = Masse

5.3. 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.

5.4. 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
```

6. Manipulation avec le Raspberry pi 3

Avant de commencer la programmation du robot, il a fallu tout d'abord installer un système d'exploitation optimisé pour le Raspberry pi. Nous avons choisi d'utiliser la version lite de la distribution Linux Raspbian. En effet, celle-ci est très légère, possède une grande communauté en cas de problème et inclut des outils de programmations intégrés nativement comme Python. Elle convenait donc parfaitement pour notre utilisation.

Afin de pouvoir contacter le Raspberry pi, nous nous sommes reliés au réseau de l'école via WiFi depuis celui-ci en paramétrant une IP statique. Ensuite nous avons pu accéder directement au Raspberry pi grâce au protocole SSH (activable en créant un dossier ssh et un fichier ssh.ssh sur la partition /boot/). Pour nous connecter dessus depuis nos machines Windows, nous avons dû installer Putty, il nous a ensuite suffi d'entrer l'adresse IP et le numéro du port utilisé par SSH (22) pour établir la connexion.

Afin d'envoyer nos codes sur le Raspberry pi et les récupérer, nous avons utilisé le logiciel

WinSCP (pour les machines Windows) qui propose une interface permettant l'échange sécurisé de fichiers avec une machine distante.

7. 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.

8. 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 et tout le groupe a donc pu travailler en parallèle sur le projet ce qui a amélioré la gestion générale de celui-ci et a permis de gérer les différentes versions du robot baliseur.
Nous avons également utilisé ces outils comme « journal de bord » pour le projet.
L'utilisation de Github pourra nous permettre entre autres de partager notre projet à une plus large communauté.
(Lien : https://github.com/l-Legacy-l/Projet-nano-ordinateur)

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

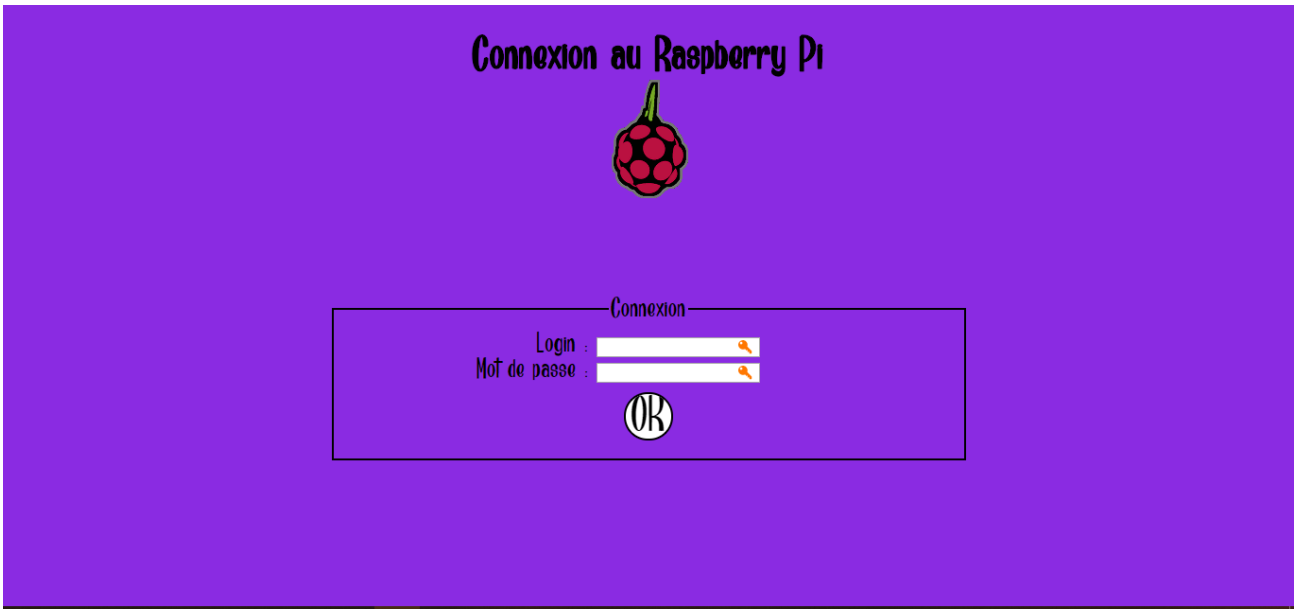
9. Contrôle du robot

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.

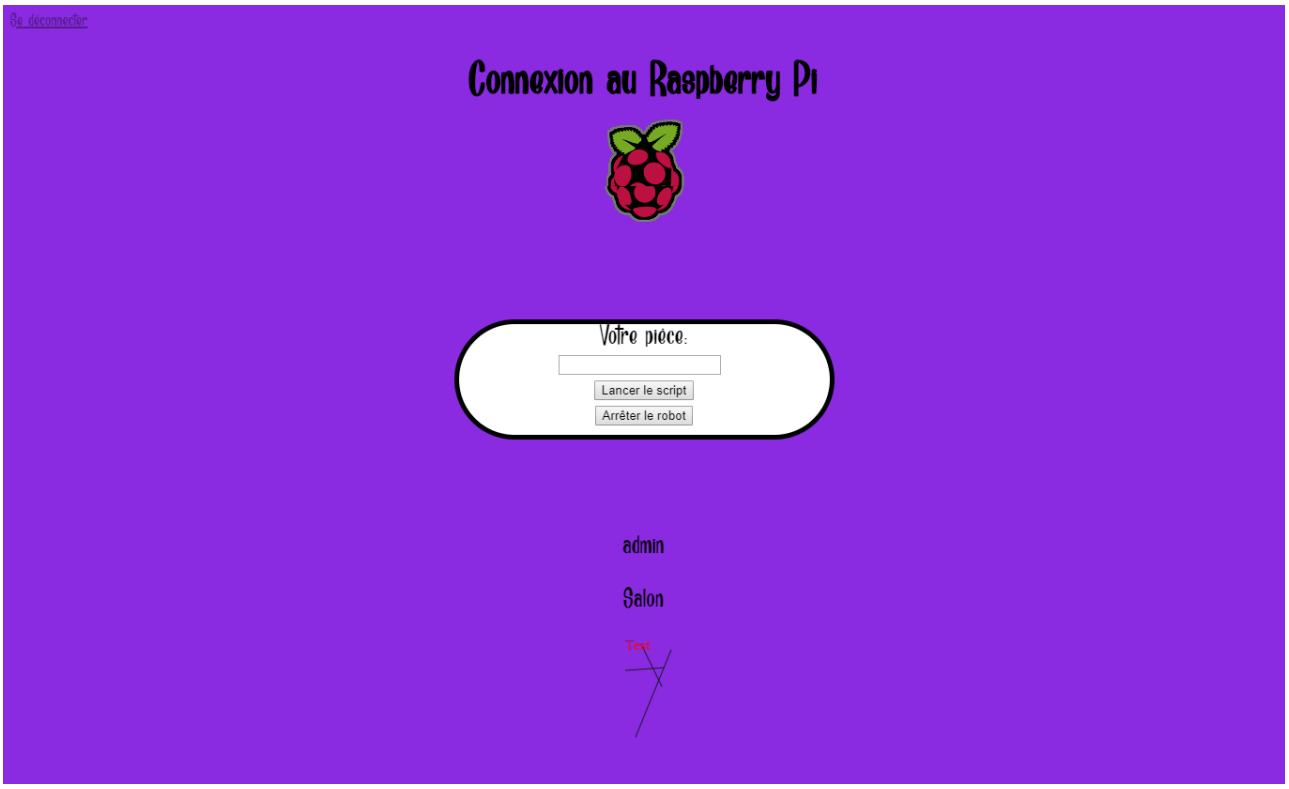
Nous avons également installé MySQL et PHP sur celui-ci.

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 sans pouvoir aller consulter celle des autres.



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

10. Algorithme de programmation

10.1. Programmation du robot en pseudo-code

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

10.2. Algorithme de déplacement

Nous avons créé un script Python utilisant la librairie du Piconzero pour déplacer le robot le long des parois. Lorsque le robot est posé à terre et que le script est exécuté via le site PHP, il se déplace jusqu'à se retrouver face à un mur. Ensuite grâce aux deux capteurs, le robot commence à faire son tour le long des parois.

Nous n'avons pas programmé de fonction permettant au robot de se localiser dans la pièce et de s'arrêter après avoir fait un tour complet. En effet, la fonction de tracking a été implémentée en parallèle avec la fonction de déplacement et elles ne communiquent pas ensemble.

10.3. Algorithme de tracking

Pour le tracking nous avons utilisé les tachymètres placés sur les roues du robot pour calculer les déplacements et les angles que le robot prend. Cependant, ces capteurs ne sont pas extrêmement précis et nous avons dû les calibrer nous-mêmes, car nous n'avons pas trouvé de documentations sur ces capteurs. Lors de chaque ligne droite, le robot compte le nombre de pas et les transforme en « mètres ». Pour les angles, le calcul est encore plus approximatif, car nous n'avons pas d'autres possibilités. De ce fait, nous avons décidé de prendre une certaine marge que nous considérons comme étant un angle droit, car ce cas est le plus courant.

10.4. Génération du fichier SVG

Le fichier généré par le script de tracking est ensuite passé dans un autre script (qui est exécutable depuis le script web) permettant de générer un SVG qui sera ensuite utilisé par le site web. Le site web renomme le fichier pour lui donner le nom choisi par l'utilisateur (la pièce indiquée) . Le fichier contient aussi du texte reprenant des données de bases comme la distance parcourue par le robot.

11. 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 d'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.

12. 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 concentrés 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.

13. 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, nous 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.

14. Bibliographie

Admin. Piconzero – Intelligent Robotics for Raspberry Pi [en ligne].
https://4tronix.co.uk/blog/?p=1224
Consulté le 26 mars 2018.

Lucien Bachelard. pj2-hc-sr04-utilisation-avec-picaxe [en ligne].
https://www.gotronic.fr/pj2-hc-sr04-utilisation-avec-picaxe-1343.pdf
Consulté le 26 mars 2018.

15. Annexe

15.1. Programmation du robot

Mapper.py

```
1. #!/bin/python
2.
3. # Import des librairies
4. import piconzero as pz
5. import time
6. from Bluetin_Echo import Echo
7. from threading import Thread
```

```
8.
9.  # Definition des pins
10. TRIGGER_PIN1 = 27
11. ECHO_PIN1 = 22
12. TRIGGER_PIN2 = 18
13. ECHO_PIN2 = 17
14.
15. # Definition de la constante physique vitesse du son pour permettre le calcul de la
    distance entre capteur/obstacle
16. speed_of_sound = 315
17.
18. # Definition du nombres d echantillon a chaque prise de donnee du capteur (moyenne si
    plusieur)
19. samples = 1
20.
21. # Definition des distances min et max entre robot et mur pour l'algorithme de
    redressement de trajectoire
22. gapmin = 20
23. gapmax = 30
24.
25. # Vitesse du robot
26. speed = 100
27.
28. # Le robot ne suis pas encore un mur au depart
29. onTrack = False
30.
31. pz.init( )
32. while True:
33.
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()
38.
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)
44.
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()
55.
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 :
65.
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()
91.
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)
101.
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
106.
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)
144.
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)
165.
166.                 else :
167.                     pz.forward(speed)
168.
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")
171.
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
183.
184.             time.sleep(0.1)
```

svgGenerator.py

```
1.  #!/bin/python
2.
3.  import re
4.  import svgwrite
5.  import math
6.
7.  # Ratio utilisé pour les angles (pas/degres)
8.  ratioAngle = 90/50
9.  # Ratio utilisé pour les lignes (metres/pas)
10. ratioLigne = 10/2000 * 1000
11.
12. # Creation d'un fichier svg
13. dwg = svgwrite.Drawing('test.svg')
14.
15. # Initialisation des variables
16. angle = 0
17. ligne = 0
18. xorg = 0
19. yorg = 0
20. perim = 0
21.
22. # Ouverture d'un fichier d'input
23. with open("output_Perfect", "r") as f :
24.     # Pour chaque ligne
25.     for line in f :
26.         # Si la ligne contient un angle
27.         if re.match("^D", line) is not None :
28.             # Trouver la valeur de l'angle en pas
29.             # a = re.findall("([0-9]+)", line)
30.             # # Transformer l'angle en degres
31.             # a = int(a[0]) * ratioAngle
32.
33.             # Toujours un angle droit
34.             a = 90
35.
36.             # Ajoute l'angle à l'angle total
37.             angle += a
38.         elif re.match("^G", line) is not None :
39.             # Trouver la valeur de l'angle en pas
40.             # a = re.findall("([0-9]+)", line)
41.             # # Transformer l'angle en degres
42.             # a = int(a[0]) * ratioAngle
43.
44.             # Toujours un angle droit
45.             a = 90
46.
47.             # Ajoute l'angle à l'angle total
48.             angle -= a
49.
50.         # Si la ligne contient une distance
51.         elif re.match("^L", line) is not None :
52.             # Trouver la valeur de la distance en pas
```

```
            ligne = re.findall("([0-9]+)", line)
            # Transformer la ligne en metres
            ligne = int(ligne[0]) * ratioLigne
            print(ligne)
            perim += ligne

            # Calcul de la nouvelle position en utilisant socathoa
            x = xorg + math.sin(angle * math.pi/180) * ligne
            y = yorg + math.cos(angle * math.pi/180) * ligne
            print("x: " + str(x) + " y: " + str(y))

            # Ajout de la ligne allant de la dernière postion vers la nouvelle position au svg
            dwg.add(dwg.line((math.fabs(xorg), math.fabs(yorg)), (math.fabs(x), math.fabs(y)), stroke=svgwrite.rgb(10, 10, 16, '%')))

            # Enregistrement des anciennes valeurs
            xorg = x
            yorg = y

        else :
            # Si la ligne ne contient ni un angle ni une ligne on affiche une erreur
            print("Le fichier n'est pas correcte")
            exit()
    dwg.add(dwg.text(str(perim) + " pas", insert=(10, 20)))
    dwg.add(dwg.text(str(perim * ratioLigne) + " metres", insert=(10, 40)))

# Sauvegarde du fichier svg
dwg.save()
```

15.2. Site web

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Raspberry Pi</title>
        <link rel="stylesheet" href="Style.css"/>
    </head>
    <body>
        <h1>Connexion au Raspberry Pi</h1>
        <div id="img">
            <img src="Images/logo.gif" >
        </div>
        <!--Formulaire de connexion Ã  l'interface du lancement du script-->
        <form action="Connexion.php" method="POST">
            <fieldset>
            <legend>Connexion</legend>
            Login : <input  type="text" name="login" required ><br>
            Mot de passe : <input id="input_password" type="password" name="password" required ><br>
            <input id="button" type="submit" value="OK"  >
            </fieldset>
        </form>

```

```
    </body>
</html>
```

Style.css

```
body
{
    background-color: blueviolet;
    font-family: "BrainFish", sans-serif;
}
/*Police*/
@font-face
{
    font-family: "BrainFish";
    src: url('Police/Brainfish_PersonalUseOnly.ttf');
}

h1
{
    font-size: 50px;
    text-align: center;
    margin-bottom: 0px;
}
#img
{
    text-align: center;
    margin-bottom: 100px;
}
p
{
    text-align: center;
    font-size:30px;
}
#svg
{
    text-align:center;
    width:300px;
    height:100px;
    margin:0 auto;
}
#svg img
{
    width:3 em;
    height:3 em;
    margin-left:130px;
}

form
{
    width: 50%;
    margin-left: auto;
    margin-right: auto;
    font-size: 30px;
    text-align: center;
}

fieldset
{
    border-color: black;
}
#bonjour
{
```

```
58.     font-size: 30px;
59. }
60.
61. #header
62. {
63.     font-size: 20px;
64. }
65. #button
66. {
67.     background-color: white;
68.     border-radius: 50px;
69.     font-size: 50px;
70.     margin-top: 10px;
71.     font-family: "BrainFish", sans-serif;
72.     border-color: black;
73. }
74. #input_password
75. {
76.     margin-right: 63px;
77. }
78.
79. #div_script
80. {
81.     text-align:center;
82.     font-size:50px;
83.     border: 5px solid black;
84.     margin-left:35%;
85.     margin-right:35%;
86.     background-color: white;
87.     border-radius:500px;
88. }
89.
90. #submit1
91. {
92.     margin-bottom:10px;
93. }
```

Connexion.php

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>Raspberry Pi</title>
5.          <link rel="stylesheet" href="Style.css"/>
6.      </head>
7.      <body>
8.          <div id="header">
9.              <a href="Logout.php">Se déconnecter</a>
10.             </br>
11.             <?php
12.                 session_start();
13.                 //On enregistre la session
14.                 $_SESSION['login'] = $_POST['login'];
15.                 //Chiffrement du mdp
16.                 $mdp = sha1($_POST['password']);
17.                 setcookie('pseudo', $_SESSION['login'], time() + 365*24*3600, null, null
    , false, true);
18.
19.                 try
20.                 {
```

```
21.                     //On se connecte à la BDD et on check si les données entrées dans le
    formulaire sont conforment à ceux de la BDD
22.                     $bdd = new PDO('mysql:host=localhost;dbname=raspberrypi;charset=utf8
    ', 'root', 'rasp789456');
23.                     $req= $bdd ->prepare('SELECT login,password FROM users WHERE login =
    :login && password = :password');
24.                     $req-
    >execute(array('login' => $_SESSION['login'], 'password' => $mdp));
25.                     $donnees = $req->fetch();
26.
27.                     //Si la requête a bien retourné une correspondance alors on est
    connecté
28.                     if(!empty($donnees))
29.                     {
30.                         echo 'Vous êtes connecté !';
31.                         $_SESSION['password']=$mdp;
32.
33.                         //On est connecté, ont peut donc accéder à l'interface de la
    cartographie, on fait une redirection
34.                         echo "<script
    type='text/javascript'>document.location.replace('Cartographie.php');</script>";
35.                     ?>
36.
37.                     <?php
38.                     }
39.
40.                     //Aucune correspondance, mauvais login/mot de passe
41.                         else
42.                         {
43.                             session_destroy();
44.                             echo "<script
    type='text/javascript'>document.location.replace('index.php');</script>";

45.                         }
46.                 }
47.
48.                 catch (Exception $e)
49.                 {
50.                     echo 'Erreur de connexion à la BD '.$e;
51.                 }
52.             ?>
53.         </div>
```

insereCartographie.php

```
1.  <?php
2.  session_start();
3.
4.  //Si on bien indiquÃ© une piÃ¨ce dans le formulaire
5.  if(!empty($_SESSION["piece"]))
6.  {
7.      try
8.      {
9.          //Chemin pour aller stocker le svg de la cartograpie
10.         $GLOBALS['image'] = $_SESSION['login'].'/'.$_SESSION['piece'].'.svg' ;
11.         //Connexion Ã  la BDD + ajout des informations dans celle-ci
12.         $bd=new PDO('mysql:host=localhost;dbname=raspberrypi;charset=utf8','root','rasp789456
    ');
13.         $req = $bd -> prepare('INSERT INTO data VALUES (:login,:piece,:image)');
```

```
14.         $req -
    > execute(array('login' => $_SESSION['login'],'piece' => $_SESSION['piece'],'image' => $GLOBA
    LS['image']));
15.
16.         //On stocke l'image pour pouvoir l'afficher dans cartographie.php
17.         $_SESSION['image'] = $GLOBALS['image'];
18.
19.     }
20.
21.     catch(Exception $e)
22.     {
23.         echo "erreur connexion a a DB";
24.     }
25. }
26.
27. ?>
```

lireCartographie.php

```
1.  <?php
2.  session_start();
3.  try
4.  {
5.      //On va récupérer les cartographie faites par l'utilisateur connecté
6.      $bdd = new PDO('mysql:host=localhost;dbname=raspberrypi;charset=utf8', 'root', 'rasp78945
    6');
7.      $req= $bdd ->prepare('SELECT * FROM data WHERE login = :login');
8.      $req->execute(array('login' => $_SESSION['login']));
9.
10.     $donnees = $req -> fetchAll();
11.     $_SESSION['resultat'] = $donnees;
12.     /*On enregistre les nombres de lignes retourné par la requêtes pour déterminer le nombre
    de cartographie
13.     qu'on a réalisé*/
14.     $_SESSION['count'] = $req -> rowCount();
15. }
16.
17. catch(Exception $e)
18. {
19.     echo "erreur connection BD";
20. }
21. ?>
```

Logout.php

```
1.  <?php
2.  //DÃ©connexion + redirection vers la page de connexion
3.  session_start();
4.  session_destroy();
5.  echo "<script type='text/javascript'>document.location.replace('Connexion.php');</script>";
6.  ?>
```

Script.php

```
1.  <?php
2.  //Pour afficher les Ã©ventuelles erreurs php
```

```
3.  error_reporting(E_ALL);
4.  ini_set('display_errors','on');
5.
6.  session_start();
7.  //Si on est connectÃ©
8.  if($_SESSION['password'])
9.  {
10.     //ExÃ©cution du script Python pour dÃ©marrer une cartographie
11.     $command=escapeshellcmd('/usr/bin/python /var/www/html/mapper.py.'
    '.$_SESSION['login'].' '.$_POST['piece']);
12.     $output=shell_exec($command);
13.     echo $output;
14.     $_SESSION['piece'] = $_POST['piece'];
15.     //A cette instruction, la cartographie est terminÃ© --> ajoute les infos de la
    cartograpie dans la BDD
16.     require('insereCartographie.php');
17.     //Redirection pour visualiser la cartographie
18.     header('Location:Cartographie.php');
19. }
20.
21. //On est pas connectÃ©, redirection
22. else
23. {
24.     echo "<script type='text/javascript'>document.location.replace('index.php');</script>";
25. }
26. ?>
```

stopScript.php

```
1.  <?php
2.  session_start();
3.  //Pour voir les erreurs
4.  ini_set('display_errors','on');
5.  error_reporting(E_ALL);
6.
7.  //Si on est connectÃ©
8.  if($_SESSION['password'])
9.  {
10.     //Execution de commande pour arrÃªter le robot
11.     $command=escapeshellcmd('/usr/bin/python /var/www/html/stop.py && /usr/bin/sudo
    /usr/bin/killall python && /usr/bin/python /var/www/html/stop.py');
12.     $output=shell_exec($command);
13.     echo $output;
14. }
15. //Redirection pour afficher sa/ses cartographies
16. echo "<script
    type='text/javascript'>document.location.replace('Cartographie.php');</script>";
17.
18. ?>
```

Raspberrypi.sql

```
1.  SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
2.  SET AUTOCOMMIT = 0;
3.  START TRANSACTION;
4.  SET time_zone = "+00:00";
5.  --
```

```sql
-- Base de données :  `raspberrypi`
CREATE DATABASE IF NOT EXISTS `raspberrypi` DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci;
USE `raspberrypi`;
-- -- --------------------------------------------------------
-- Structure de la table `data`-

DROP TABLE IF EXISTS `data`;
CREATE TABLE IF NOT EXISTS `data` (
  `login` varchar(255) NOT NULL,
  `piece` varchar(255) NOT NULL,
  `image` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------
-- Structure de la table `users`
--

DROP TABLE IF EXISTS `users`;
CREATE TABLE IF NOT EXISTS `users` (
  `login` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Déchargement des données de la table `users`

INSERT INTO `users` (`Login`, `Password`) VALUES
('admin', '96fd0508f32de904e0cd8e9e4c6a62a4a3b41052'),
('fabio', '96fd0508f32de904e0cd8e9e4c6a62a4a3b41052'),
('timothee', '96fd0508f32de904e0cd8e9e4c6a62a4a3b41052'),
('sam', '96fd0508f32de904e0cd8e9e4c6a62a4a3b41052'),
('jerome', '96fd0508f32de904e0cd8e9e4c6a62a4a3b41052');
COMMIT;
```