Systèmes intelligents
Rapport

Réalisé par : GOIRE Antoine et DUCHENE Roy

Professeur : Mme Slama et Mme Liberatore

Année d'étude : Master 1-finalité automatisation

Année académique 2018-2019.

Table des matières :

1) Description du projet

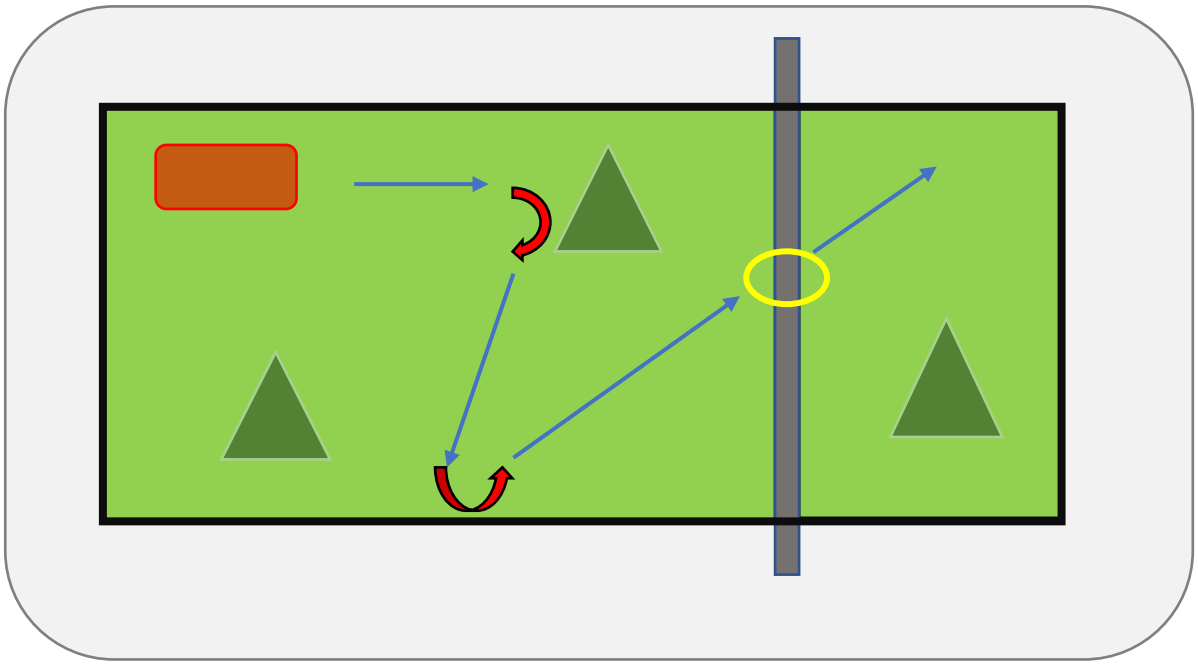
ARDUINO

Un robot tondeuse, qui circule aléatoirement dans un périmètre défini par une ligne d'une certaine couleur (exemple : le rouge). Cette ligne est détectée par le robot grâce à un détecteur de couleur RGB situé à l'avant de celui-ci. Il y a également un capteur à ultrasons qui détecte si un objet ou autre obstrue le chemin. Une fois qu'il s'en approche de X cm, il fait demi-tour et prend une autre direction. Si cet objectif est atteint c'est-à-dire en bonus, il est prévu d'ajouter une sécurité sur le robot tondeuse c'est-à-dire un capteur qui sera installé vers le sol. Quand une personne soulèvera la tondeuse cela coupera le moteur des lames ainsi que les moteurs des roues et un bip s'enclenchera également.

RASPBERRY

Détection de la matière au sol grâce à la caméra qui sera câblée sur la Raspberry. Si c'est du gazon on actionne le moteur de la coupe en revanche si ce n'est pas de l'herbe alors le moteur se coupe et un bip s'actionne. On affichera sur l'écran LCD si c'est du gazon ou pas. En bonus, nous ajouterons sur l'écran LCD la couleur du gazon, sa qualité ainsi que si le gazon est bien tondu ou non.

SCHEMA



LEGENDE

Robot tondeuse

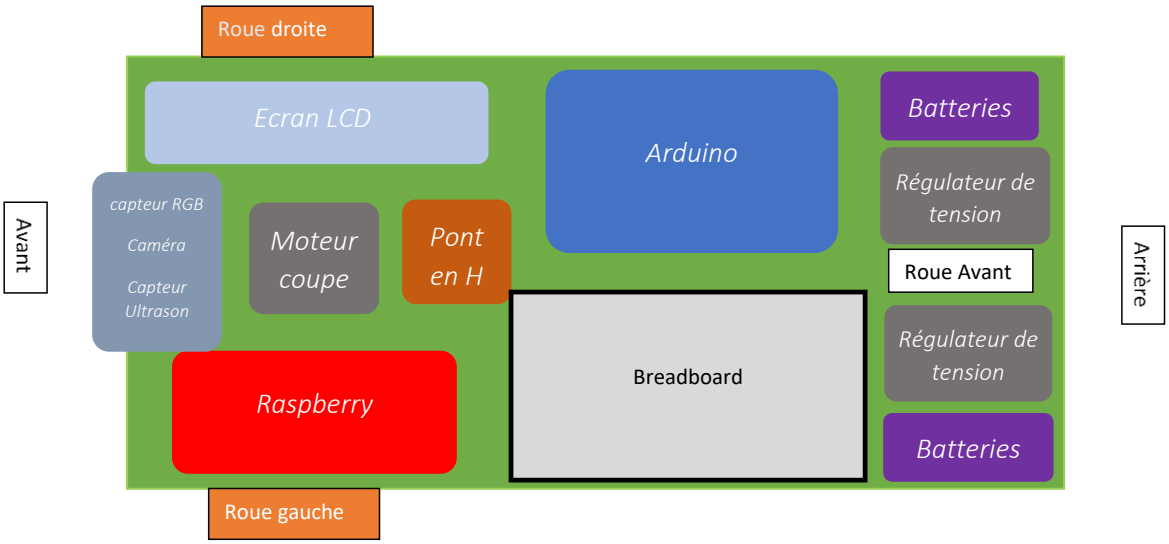
Arbre ou obstacle

Route qui traverse l'herbe (état de surface différente de celle de l'herbe).

Contour dans lequel le robot ne devra pas sortir.

Le robot déclenche un BIP et arrête le moteur de coupe.

PREMIER PLAN DU ROBOT TONDEUSE



2) Description du matériel nécessaire

Nombre	Matériel	Code article	Lien	Prix
1	Module à détection US HC-SR04A	35750	https://www.gotronic.fr/art-module-a-detection-us-hc-sr04a-27740.htm	3,90 €
1	Capteur de couleur SEN0101	31752	https://www.gotronic.fr/art-capteur-de-couleur-sen0101-19374.htm	9,90 €
2	Kit roue+moteur 6VCC-100t/min	33912	https://www.gotronic.fr/art-kit-roue-moteur-23569.htm	2x 5,50€
3	Régulateurs ajustables 1,25à30VCC GT134	35189	https://www.gotronic.fr/art-regulateur-ajustable-1-25-a-30-vcc-gt134-26094.htm	3x2,50€
1	Commande de 2 moteurs SBC-motordriver2	35683	https://www.gotronic.fr/art-commande-de-2-moteurs-sbc-motodriver2-27418.htm	6,90 €
2	Module relais 5 V GT1080	35226	https://www.gotronic.fr/art-module-relais-5-v-gt1080-26130.htm	2x3,80€
1	Kit GPIO pour raspberry PI PI018	35118	https://www.gotronic.fr/art-kit-gpio-pour-raspberry-pi-pi018-25862.htm	10,90 €
1	moteur pour couper l'herbe	25350	https://www.gotronic.fr/art-moteur-standard-rm1a-12005.htm	3,20€
1	écran LCD	34940	https://www.gotronic.fr/art-afficheur-lcd-i2c-2x16-caracteres-25650.htm	9,90€
1	piezzo	05496	https://www.gotronic.fr/art-capsule-piezoelectrique-dp035f-3856.htm	1,20€
1	caméra raspberry	35368	https://www.gotronic.fr/art-camera-8-mpx-pour-raspberry-rb-camera-v2-26774.htm	36,50€
1	raspberry	35790	https://www.gotronic.fr/art-carte-raspberry-pi3-b-27826.htm	43,95€
1	arduino ATmega	25951	https://www.gotronic.fr/art-carte-arduino-mega-2560-12421.htm	38.90€
1	roulette (avant)	/	/	Pack
2	support de piles	/	/	Pack
X	piles	/	/	Pack

Module à détection US HC-SR04A

Capteur de couleur SEN0101

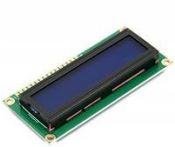
Kit roue+moteur 6VCC-100t/min

Régulateurs ajustables 1,25à30VCC GT134

Commande de 2 moteurs SBC-motordriver2

Module relais 5 V GT1080

Kit GPIO pour Raspberry PI PI018



Écran LCD

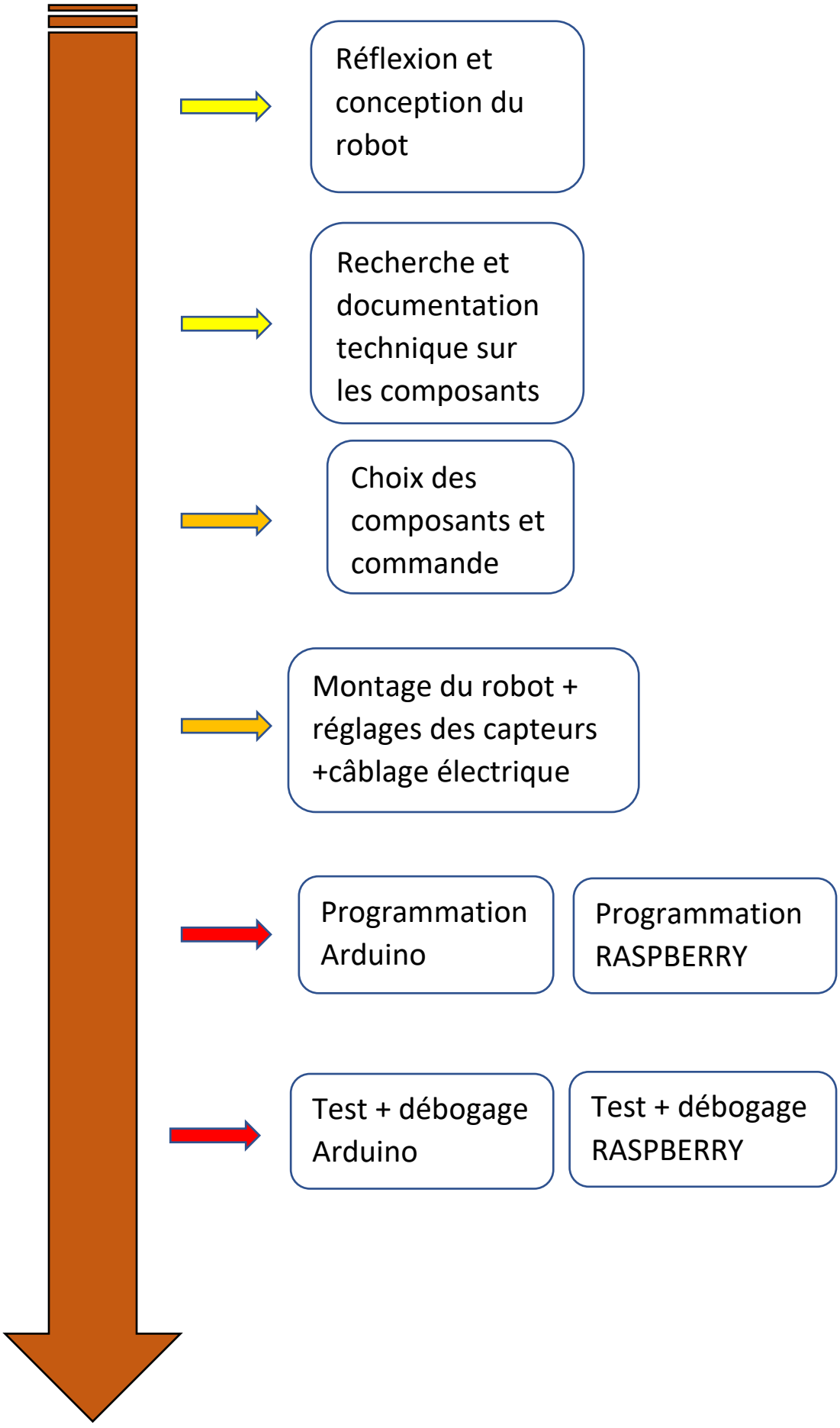
Arduino

Raspberry

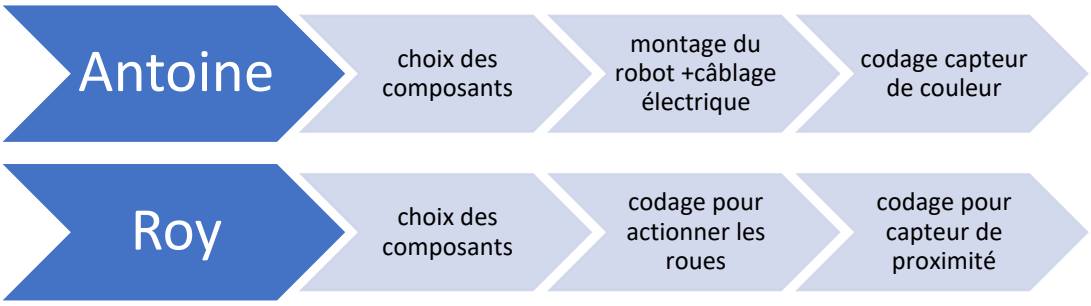
Roulette avant du robot

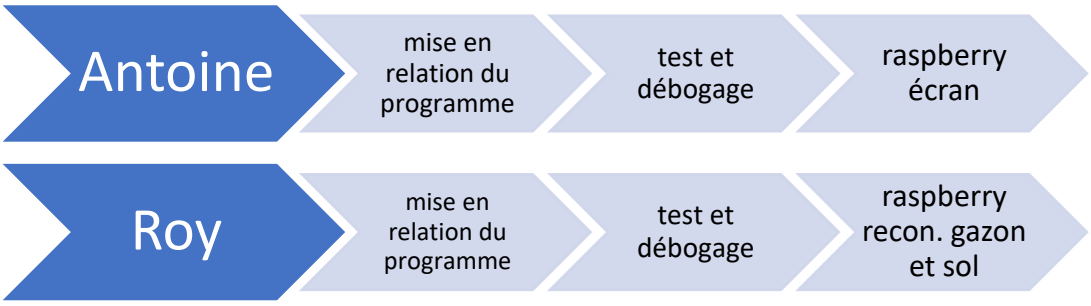
Moteur pour couper l’herbe

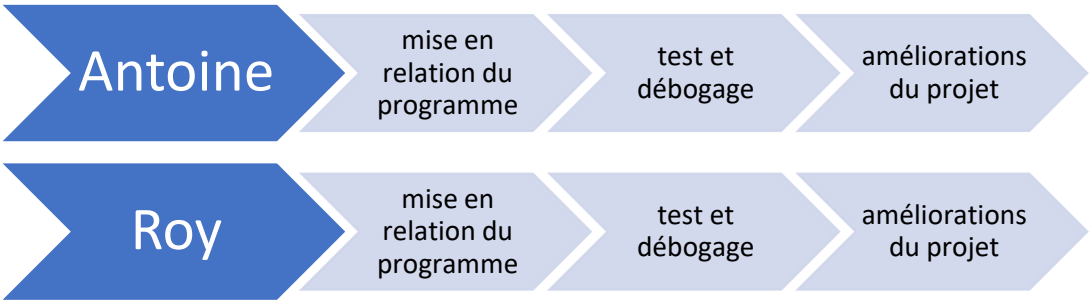
3) Diagramme du projet



4) Répartition des tâches au sein du groupe







5) Time Line avec étapes de validations

05/02	06/02	12/02	13/02
Réflexion robot	Réflexion robot	Préparation de la présentation du robot	Préparation de la présentation du robot
Réflexion robot	Réflexion robot	Préparation de la présentation du robot +cahier des charges	Préparation de la présentation du robot +cahier des charges
19/02	**20/02**	**26/02**	**27/02**
Choix du matériel +cahier des charges	Choix du matériel + cahier des charges	Présentation du robot + passage de la commande	Cahier des charges
Choix du matériel +cahier des charges	Choix du matériel +cahier des charges	Présentation du robot + passage de la commande	Cahier des charges
12/03	**13/03**	**19/03**	**20/03**
Codage capteur de couleur	Codage pour actionner les roues	Mise en relation programme + test et débogage	Test et débogage
Codage du capteur de proximité	Codage pour actionner les roues	Mise en relation programme + test et débogage	Test et débogage
27/03	**03/04**	**24/04**	**08/05**
Raspberry écran LCD	Communication Arduino Raspberry	Mise en relation du programme + Test et débogage	Test et débogage
Raspberry reconnaissance gazon et sol	Raspberry reconnaissance gazon et sol	Mise en relation du programme + Test et débogage	Test et débogage
15/05			
Améliorations (Mise au propre du code + commentaires etc)			
Améliorations			

(Mise au propre du code + commentaires etc)			
Vacances Carnaval 04/03 => 08/03			
Montage robot	Câblage électrique +réglages des capteurs	Temps pour s'il y a un retard à rattraper	
/	/	Temps pour s'il y a un retard à rattraper	
Vacances Pâques 08/04 => 19/04			
Temps pour s'il y a un retard à rattraper			
Temps pour s'il y a un retard à rattraper			

6) Etat de l'art de l'existant

Code pour moteur driver Arduino

Code utile pour le déplacement de notre robot tondeuse. Le fonctionnement de ce programme fonctionne bien. Cependant, dans la suite de notre programmation nous avons utilisé des fonctions qui permettent d'éclaircir le programme et de le rendre plus compréhensible (Voir annexe n°1).

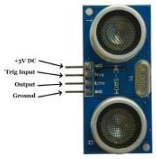
Code utile pour le capteur à ultrason

Permet au robot d'éviter les obstacles présents dans son environnement de travail.
Le fonctionnement du module est le suivant :

Il faut envoyer une impulsion niveau haut (à + 5v) pendant au moins 10 µs sur la broche '**Trig Input**' ; cela déclenche la mesure. En retour la sortie '**Output**' ou '**Echo**', va fournir une impulsion + 5v dont la durée est proportionnelle à la distance si le module détecte un objet.

Afin de pouvoir calculer la distance en cm, on utilisera la formule suivante :

Distance = (durée de l'impulsion (en µs) / 58.



(Voir annexe n°2).

Code utile pour le capteur RGB

Permet de détecter les limites établies par une ligne de couleur. Le programme est très simple à comprendre. Cependant, ce programme n'est pas assez complet pour l'application que nous voulons en faire. Il est donc impératif de trouver ou d'améliorer le programme afin qu'il convienne à nos exigences (Voir annexe n° 3).

Stratégie de tonte

A première vue, la stratégie trouvée est une bonne stratégie de tonte. Il y a tout de même quelques inconvénients dans cette stratégie. Premièrement, un robot tondeuse comme nous pouvons l'acheter dans un commerce n'a pas de stratégie de tonte. C'est-à-dire qu'elle a un parcours qui est aléatoire.

Deuxièmement, dans le plan concret de notre robot il est impossible de faire un demi-tour sur place avec un tel angle de rotation. Nous avons donc mis un angle aléatoire pour lequel le robot se déplace sans se heurter dans le mur (Voir annexe n°4).

Raspberry

-Nous avons utilisé ce lien [1] afin d'installer les bibliothèques utiles pour la reconnaissance de texture.

-Nous nous sommes basés sur ce programme [2] pour effectuer la reconnaissance de texture. À la suite d'un manque de temps et certains problèmes rencontrés lors de la réalisation du projet, nous n'avons pas su tester sur ordinateur le programme. Néanmoins, nous avons fait ci-dessous une description complète du programme

-Ce dernier lien [3] est un programme de reconnaissance de couleur sur lequel nous nous sommes appuyés. Des modifications ont dû être apportées afin de correspondre à nos besoins.

[1] https://www.framboise314.fr/i-a-realisez-un-systeme-de-reconnaissance-dobjets-avec-raspberry-pi/

[2] https://gogul09.github.io/software/texture-recognition

[3] https://stackoverflow.com/questions/50178248/detect-color-in-the-middle-of-image-opencv-python

7) Déroulement du projet au fil des semaines

05/02 au 27/02 :

- réflexion robot.

-présentation robot.

-choix du matériel.

-conception du cahier des charges.

-passage de la commande.

Aucun retard sur le Time Line.

04/03 au 08/03 :

- Montage robot.

-câblage électrique.

- premier test moteur de coupe.

-premier test capteur de proximité + codage des roues.

-Amélioration du programme du capteur de proximité et du codage des roues.

Aucun retard sur le Time Line.

11/03 au 15/03 :

-affinement du code pour le capteur de proximité + codage roues :

Problème rencontré : le robot ne se déplace pas droit, il dévie de sa trajectoire du au poids qui n'est pas équilibré sur tout le châssis du robot.

Solution : on peut faire varier la vitesse des roues du robot. On a donc joué sur cela pour faire avancer le robot sur une trajectoire qui est droite.

-Amélioration du câblage électrique.

Problème rencontré : le pont en H ne fonctionnait pas.

Solution : On devait alimenter en 5 V une entrée sur le pont en H afin de faire tourner les roues en sortie du pont en H.

Retard codage du capteur RGB.

18/03 au 22/03 :

-Câblage du capteur RGB sur Arduino UNO et test d'un premier programme (programme fonctionnel).

-Changement d'Arduino UNO à un Arduino MEGA : modifications des pins (câblage et programmation).

-Test pour introduire le moteur de coupe dans le programme du capteur RGB.

-Changement de câblage du capteur de proximité et des roues (Arduino UNO à l'Arduino MEGA).

-Rapport : état d'avancement de la conception du robot.

Retard mise en relation entre le capteur RGB et le capteur de proximité ainsi que le codage des roues.

Retard également au niveau du test et débogage de la programmation Arduino.

25/03 au 29/03 :

-Mise en relation du codage du RGB avec le codage du capteur de proximité et des roues.

-Tests du programme plus débogage.

Retard Raspberry écran LCD.

01/04 au 05/04 :

-Raspberry installation open cv.

- test allumer la caméra.

-test programme reconnaissance d'objets.

Retard Raspberry écran LCD.

Retard Raspberry communication Arduino – Raspberry.

Vacances de Pâques :

-Suite du montage robot, placement du capteur RGB, placement de la protection du robot, …

-rédaction du rapport.

-test d'un programme de reconnaissance.

22/04 au 26/04 :

-Rédaction du rapport.

- installation de différentes librairies pour la reconnaissance texture sur Raspberry + test d'un programme de reconnaissance.

Problème rencontré : la Raspberry beug et nous devons réinitialiser et installer un nouveau Raspbian dessus.

Solution : utiliser l'ordinateur pour faire ce genre de traitement car celui-ci est plus performant. Le Raspberry est alors utilisé pour faire des captures d'images et il les enverra directement sur le pc pour effectuer la reconnaissance de texture.

29/04 au 03/05 :

-Rédaction du rapport

- test programme pour récupérer les valeurs du RGB (red green blue).

-Test et débogage écran sur Raspberry + programme python.

06 /05 au 10/05 :

-Rédaction du rapport

-Raspberry code pour caméra faire une capture d'image.

-Raspberry code pour détecter la couleur de l'image.

- Raspberry sortir sur le LCD si herbe verte ou pas herbe verte.

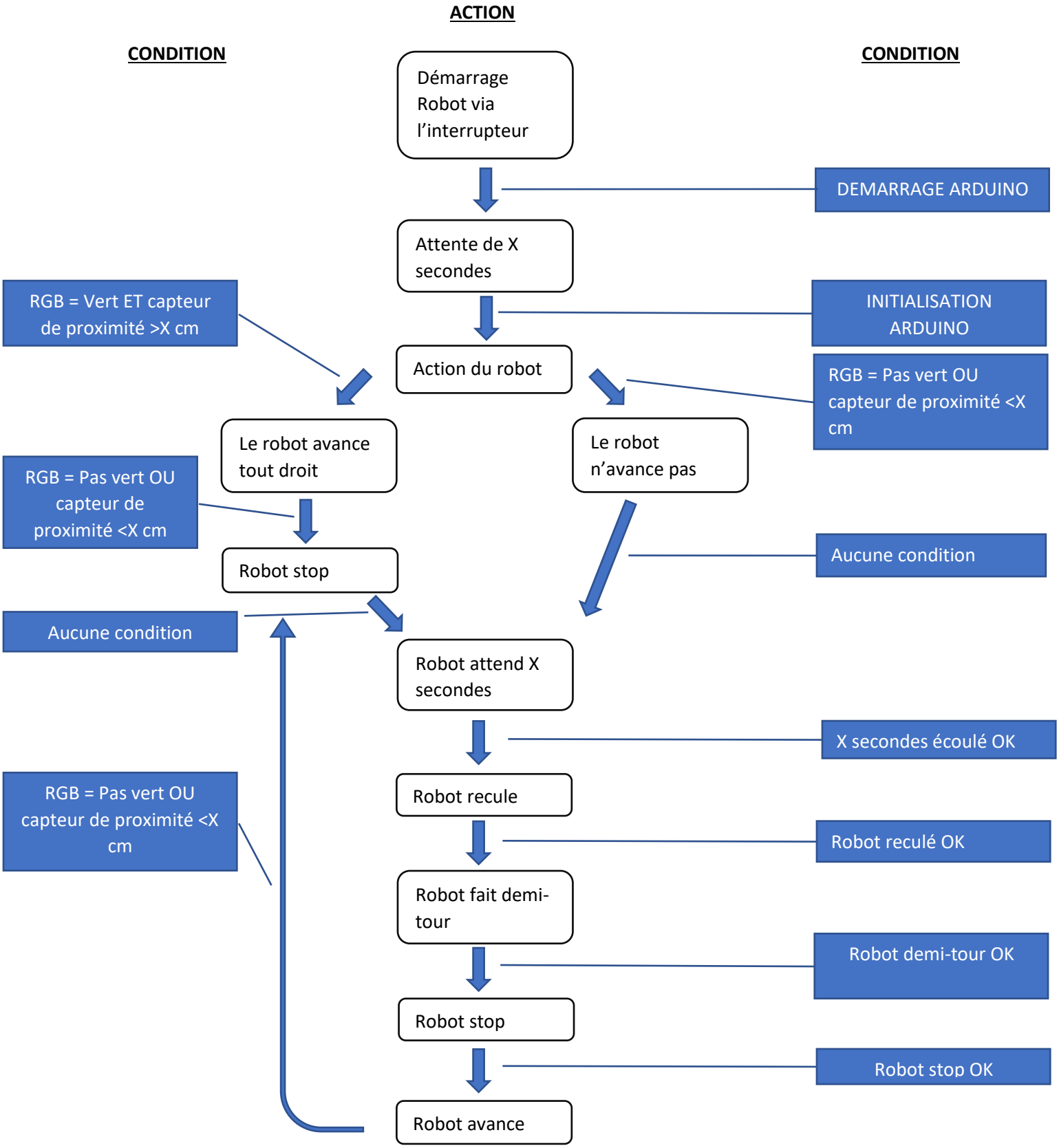
- Raspberry activer un PIN gpio en fonction de la couleur.

Problème rencontré : Les programmes réalisés en python ne s'exécutaient pas.

Solutions : Installation des bibliothèques en python 3 et non en python.

8) PARTIE 1 : Arduino

Diagramme général du déplacement du robot.



Mise en forme du programme Arduino

DEFINE : Définition des numéros de pins (voir schéma ci-dessous)

VOID SETUP : Définition des entrées et sorties des pins (Voir github programme Arduino)

VOID LOOP :

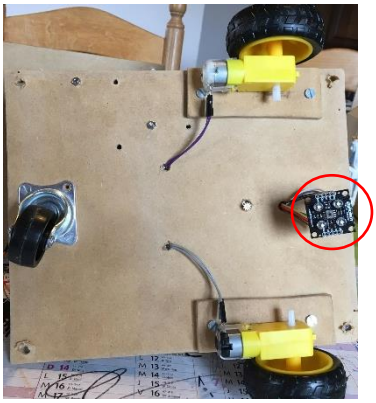
1- Appel de la fonction du calcul de distance (3)
2- Déplacement du robot grâce aux appels des différentes fonctions (3,4,5,6,7,8)
3- Fonction du calcul de distance
4- Fonction de la marche avant
5- Fonction de la marche arrière
6- Fonction du demi-tour
7- Fonction du stop
8- Fonction du calcul pour les couleurs RGB

Montage du robot

La première étape lors du montage du robot a été de découper le châssis de celui-ci qui est en bois comme on peut le voir. Ensuite nous avons fixé une roulette sur l'arrière du robot et à l'avant les moteurs avec les roues pour faire avancer, reculer, demi-tour au robot.

Ensuite, nous avons placé tous les composants dessus afin de voir la place disponible sur le châssis du robot mais aussi de placer au mieux ces composants.

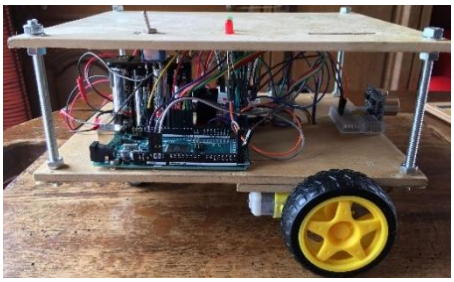
Une fois les composants placés sur le châssis, nous avons enfin pu commencer à câbler électriquement les différents composants.



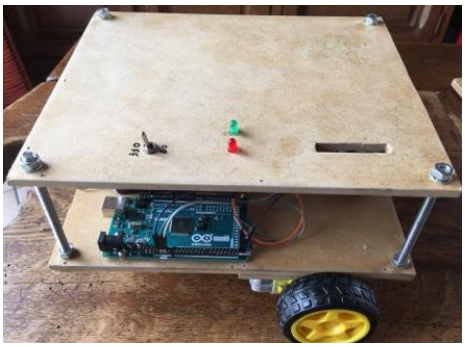
Le capteur RGB a été ajouté par la suite comme on peut le voir en rouge.

Ici, nous avons conçu une nouvelle plaque de bois pour mettre par-dessus le châssis du robot afin de cacher les fils du câblage électrique.

Des trous ont été réalisés afin de faire passer 2 leds par-dessus ainsi qu'un interrupteur et l'écran LCD du Raspberry pi.



Allure générale du robot tondeuse.

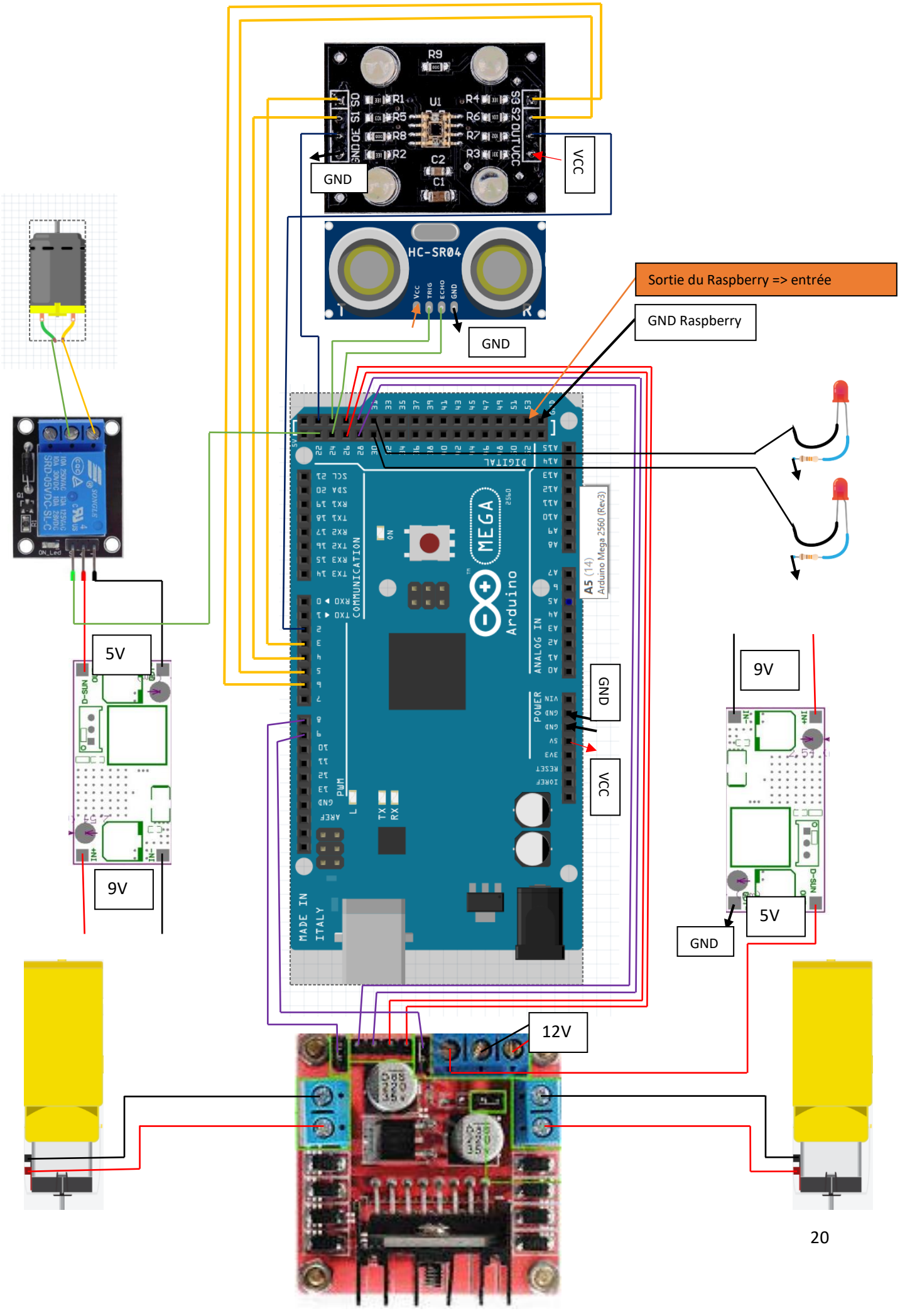


Sur le dessus du robot nous pouvons voir des leds et interrupteur qui, lui, sert de on/off pour couper l'Arduino.

Néanmoins, il reste encore quelques petites modifications à apporter au robot avant la fin du montage comme on peut le voir ci-dessous.

Schéma électrique et câblage de l'Arduino

Voici ci-dessous le schéma de câblage concernant Arduino ainsi que le reste des composants tels que le moteur de coupe, le relais, les régulateurs de tensions, …


R9
R1
U1
R4
R5
R6
R8
R7
R2
C2
R3
C1
GND
VCC
HC-SR04
Vcc
TRIG
ECHO
GND
GND
Sortie du Raspberry => entrée
GND Raspberry
5V
9V
GND
VCC
9V
5V
GND
12V
MEGA
Arduino
DIGITAL
ANALOG IN
COMMUNICATION
POWER
PWM
MADE IN ITALY
A5 (14)
Arduino Mega 2560 (Rev3)

9) PARTIE 2 : Raspberry + reconnaissance

Mise en forme du code pour la caméra (RGB)

Code Python

Importation des bibliothèques

Définition des GPIO du Raspberry vers Arduino

Configuration de la caméra

Configuration des Pins du LCD

While :

- Capture d'image via la caméra
- Traitement de l'image (définition du RGB)
- Condition d'affichage pour console windows, LCD, activation GPIO

Linux

Importation des bibliothèques sous Python3 :

- Time
- CV2
- Numpy
- Adafruit LCD
- PIcamera
- GPIO

Description du code pour la reconnaissance de texture

(Voir annexe n°5)

Importer le package nécessaire

```
1  import cv2
2  import numpy as np
3  import os
4  import glob
5  import mahotas as mt
6  from sklearn.svm import LinearSVC
```

Charger le jeu de données d'apprentissage

```
1  # load the training dataset
2  train_path = "dataset/train"
3  train_names = os.listdir(train_path)
4
5  # empty list to hold feature vectors and train labels
6  train_features = []
7  train_labels = []
```

- La ligne 2 est le chemin d'accès au jeu de données d'apprentissage.
- La ligne 3 obtient les noms de classe des données d'apprentissage.
- Les lignes 6-7 sont des listes vides pour contenir des vecteurs de fonctionnalités et des étiquettes.

Fonction d'extraction d'entités

```
1  def extract_features(image):
2          # calculate haralick texture features for 4 types of adjacency
3          textures = mt.features.haralick(image)
4
5          # take the mean of it and return it
6          ht_mean = textures.mean(axis=0)
7          return ht_mean
```

- La ligne 1 est une fonction qui prend une image d'entrée pour calculer la texture Haralick.
- La ligne 3 extrait les caractéristiques du Haralick pour les 4 types de contiguïté.
- La ligne 6 trouve la moyenne des 4 types de GLCM.
- La ligne 7 renvoie le vecteur de fonction résultant pour cette image qui décrit la texture.

Extraire des fonctions pour toutes les images

```
1  # loop over the training dataset
2  print "[STATUS] Started extracting haralick textures.."
3  for train_name in train_names:
4          cur_path = train_path + "/" + train_name
5          cur_label = train_name
6          i = 1
7          for file in glob.glob(cur_path + "/*.jpg"):
8                  print "Processing Image - {} in {}".format(i, cur_label)
9                  # read the training image
10                 image = cv2.imread(file)
11
12                 # convert the image to grayscale
13                 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
14
15                 # extract haralick texture from the image
16                 features = extract_features(gray)
17
18                 # append the feature vector and label
19                 train_features.append(features)
20                 train_labels.append(cur_label)
21
22                 # show loop update
23                 i += 1
```

- Ligne 4 boucles sur les étiquettes de formation que nous venons d'inclure dans le répertoire de formation.
- La ligne 5 est le chemin d'accès au répertoire actuel de la classe d'images.
- La ligne 6 contient l'étiquette actuelle de la classe d'images.
- La ligne 8 prend tous les fichiers avec. jpg comme l'extension et boucle à travers chaque fichier un par un.
- La ligne 11 lit l'image d'entrée qui correspond à un fichier.
- La ligne 14 convertit l'image en niveaux de gris.
- La ligne 17 extrait les fonctions de Haralick pour l'image en niveaux de gris.
- La ligne 20 ajoute le vecteur de fonction 13-Dim à la liste des fonctions d'entraînement.
- La ligne 21 ajoute l'étiquette de classe à la liste des classes de formation.

```
1  # have a look at the size of our feature vector and labels
2  print "Training features: {}".format(np.array(train_features).shape)
3  print "Training labels: {}".format(np.array(train_labels).shape)
```

Créer le classifieur machine learning

```
1  # create the classifier
2  print "[STATUS] Creating the classifier.."
3  clf_svm = LinearSVC(random_state=9)
4
5  # fit the training data and labels
6  print "[STATUS] Fitting data/label to model.."
7  clf_svm.fit(train_features, train_labels)
```

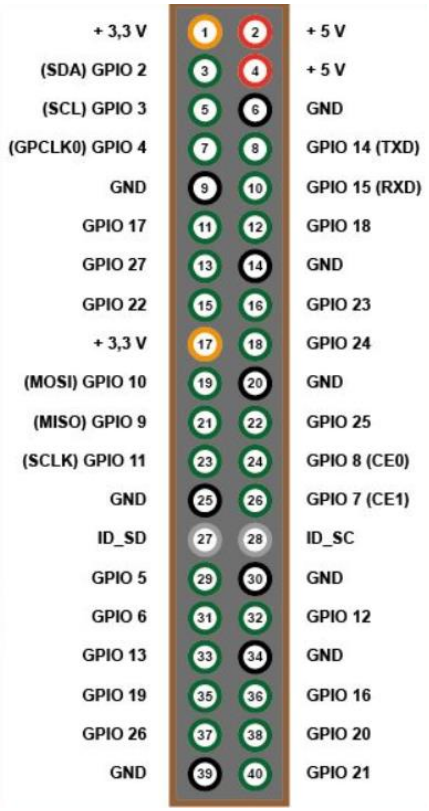
- La ligne 3 crée le classifieur Linear support vector machine.
- La ligne 7 correspond aux fonctions d'entraînement et aux étiquettes du classifieur.

Tester le classifieur sur les données de test

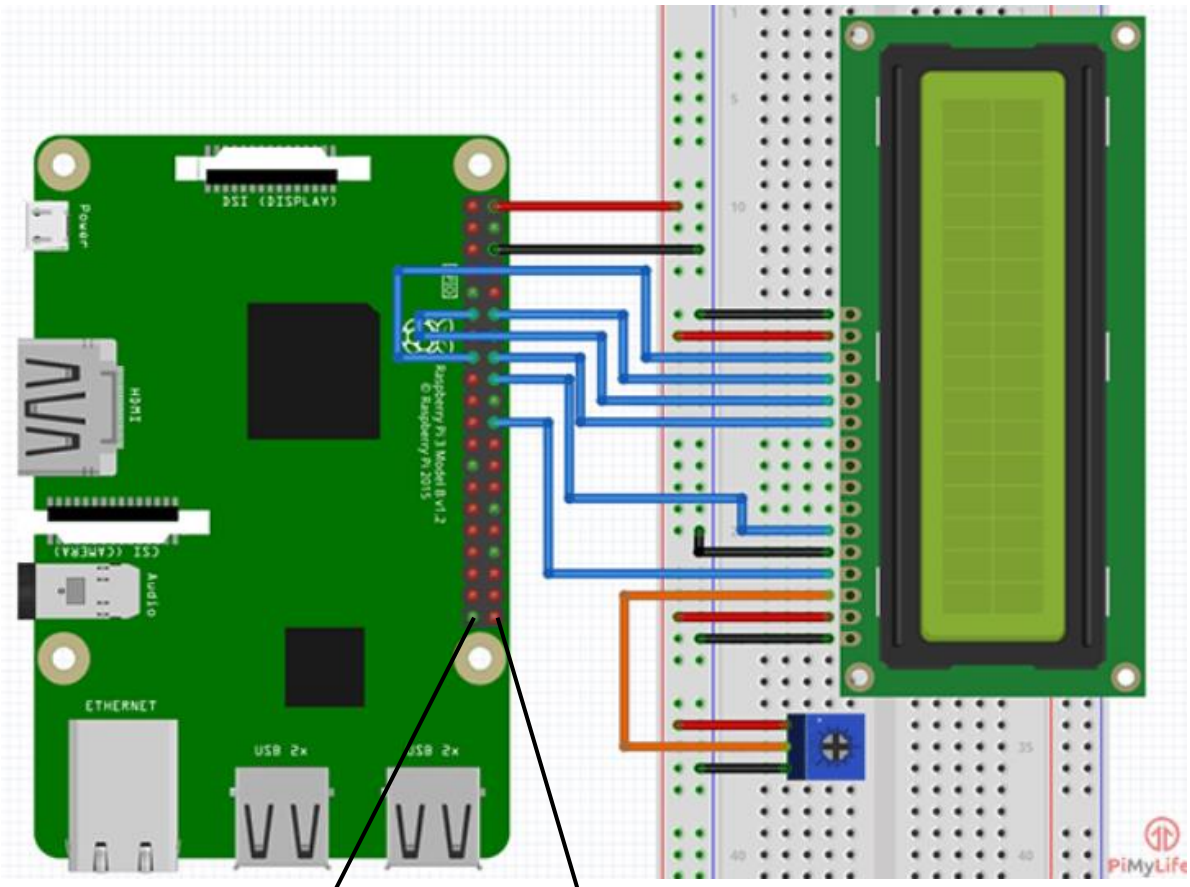
```
1  # loop over the test images
2  test_path = "dataset/test"
3  for file in glob.glob(test_path + "/*.jpg"):
4          # read the input image
5          image = cv2.imread(file)
6
7          # convert to grayscale
8          gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
9
10         # extract haralick texture from the image
11         features = extract_features(gray)
12
13         # evaluate the model and predict label
14         prediction = clf_svm.predict(features.reshape(1, -1))[0]
15
16         # show the label
17         cv2.putText(image, prediction, (20,30), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0,255,255), 3)
18
19        # display the output image
20        cv2.imshow("Test_Image", image)
21        cv2.waitKey(0)
```

- La ligne 2 obtient le chemin de données de test.
- La ligne 3 prend tous les fichiers avec l'extension. jpg et parcourt chaque fichier un par un.
- La ligne 5 lit l'image d'entrée.
- La ligne 8 convertit l'image d'entrée en image en niveaux de gris.
- La ligne 11 extrait les caractéristiques de Haralick de l'image en niveaux de gris.
- La ligne 14 prédit l'étiquette de sortie de l'image de test.
- La ligne 17 affiche l'étiquette de classe de sortie pour l'image de test.
- Enfin, la ligne 20 affiche l'image de test avec l'étiquette prévue.

Schéma électrique et câblage du Raspberry



PIN NO	Symbol	Fuction
1	VSS	GND
2	VDD	+5V
3	V0	Contrast adjustment
4	RS	H/L Register select signal
5	R/W	H/L Read/Write signal
6	E	H/L Enable signal
7	DB0	H/L Data bus line
8	DB1	H/L Data bus line
9	DB2	H/L Data bus line
10	DB3	H/L Data bus line
11	DB4	H/L Data bus line
12	DB5	H/L Data bus line
13	DB6	H/L Data bus line
14	DB7	H/L Data bus line
15	A	+4.2V for LED
16	K	Power supply for BKL(0V)



GND vers Arduino

Vers arduino Pin 53

10) **Conclusion**

En conclusion, le projet du robot tondeuse est fonctionnel malgré quelques problèmes rencontrés durant son élaboration.

Arduino
La programmation de la vitesse de chacun des moteurs nous a posée quelques soucis afin que le robot se déplace en ligne droite. Nous pensons que le problème est dû à une non-régulation de la vitesse de chacune des roues ou que les moteurs choisis ne seraient pas suffisamment puissants pour la masse de celui-ci. Il y aussi quelques soucis qui persistent avec le capteur de couleurs qui permet de délimiter l'espace dans lequel le robot tondeuse doit se déplacer, cela est dû à des problèmes de variance de luminosités.

Raspberry
La programmation de la détection des couleurs avec la pi caméra est fonctionnelle après quelques difficultés rencontrées lors de la programmation, mais ceci est la partie que nous nous étions fixée comme finalisation première du projet. Pour ce qui est de la deuxième partie nous avons rencontré beaucoup de problèmes dû au manque de ressources de la Raspberry pi sur lequel une reconnaissance de texture devient en matière de programmation et de performance de celle-ci beaucoup trop importante.
Pour ce qui est de la reconnaissance, de manière générale cela est assez complexe et demande beaucoup plus de connaissances et de temps, afin de réaliser un projet de telle envergure, et d'arriver à le finaliser au maximum.

Nous avons appris énormément de choses sur l'utilisation de la Raspberry, de l'Arduino et de la reconnaissance d'images à l'aide de open CV, ainsi que les divers langages utilisés comme le python et le C++.

Pour finaliser cette conclusion, la réalisation d'un robot tondeuse domestique n'est pas une chose facile à créer.

11) Bibliographie

https://www.robot-maker.com/shop/blog/26_Utiliser-capteur-ultrasons.html?fbclid=IwAR2cVLnEHxlicDHMF6A3GX3AQWznPh8891vAslHJ6WI_1OorDSXE4YkSik0

https://www.memorandum.ovh/tuto-arduino-utiliser-un-module-ultrason-hc-sr04/?fbclid=IwAR1ls9mw31zopeNpOXElmu1V3gniWkf6wLxeIKbYV1HN7kK2pRakWzENoDo

https://www.carnetdumaker.net/articles/mesurer-une-distance-avec-un-capteur-ultrason-hc-sr04-et-une-carte-arduino-genuino/?fbclid=IwAR2aruKkAlABbg9ZXP4kB1LmEvutT2hzZSfereeIF5wlidjCe5eB87N9OQs

https://gogul09.github.io/software/texture-recognition

https://www.google.com/search?biw=1536&bih=754&tbm=isch&sa=1&ei=9ePSXOWsAoOWapu4laAO&q=raspberry+pi+entr%C3%A9e+sortie+schema&oq=raspberry+pi+entr%C3%A9e+sortie+schema&gs_l=img.3...5555.8093..8193...0.0..0.104.1052.13j1.....1....1..gws-wiz-img.....0i24.FgnmW6JtktE#imgrc=lcodWfZ0_i-ByM:

https://wiki.mchobby.be/index.php?title=Rasp-Hack-Afficheur-LCD-Python

https://electronicshobbyists.com/raspberry-pi-camera-module-tutorial-taking-picture-and-video-recording/

https://www.raspberrypi.org/forums/viewtopic.php?t=234358

https://www.science-emergence.com/Articles/Trouver-la-plus-petite-valeur-et-lindice-dans-une-liste-de-nombres-sous-python-1/

https://www.tutorialspoint.com/python-program-to-find-maximum-and-minimum-element-s-position-in-a-list

https://stackoverflow.com/questions/2474015/getting-the-index-of-the-returned-max-or-min-item-using-max-min-on-a-list

https://www.programiz.com/python-programming/methods/built-in/max

https://www.framboise314.fr/i-a-realisez-un-systeme-de-reconnaissance-dobjets-avec-raspberry-pi/

https://stackoverflow.com/questions/50178248/detect-color-in-the-middle-of-image-opencv-python

https://assos.centrale-marseille.fr/clubrobot/sites/assos.centrale-marseille.fr.clubrobot/files/PT_annexes_E%3DM6_2013.pdf

https://www.gotronic.fr/art-capteur-de-couleur-sen0101-19374.htm

https://www.gotronic.fr/pj2-tcs3200-fr-1446.pdf

https://www.framboise314.fr/i-a-realisez-un-systeme-de-reconnaissance-dobjets-avec-raspberry-pi/?fbclid=IwAR2b0pIqOZ_JIEG-rEK6HkQ3THU4tuny3lbHrB-ApUJUBj9VGZLJEr4S2kA

https://www.createursdemondes.fr/2016/03/opencvakaze-pour-la-reconnaissance-de-formes/?fbclid=IwAR3tRO0aYdXACJdlzVbu7pB-q7bTbxnab_kVWG47FEjmfmiiQNN7KY5EQq8

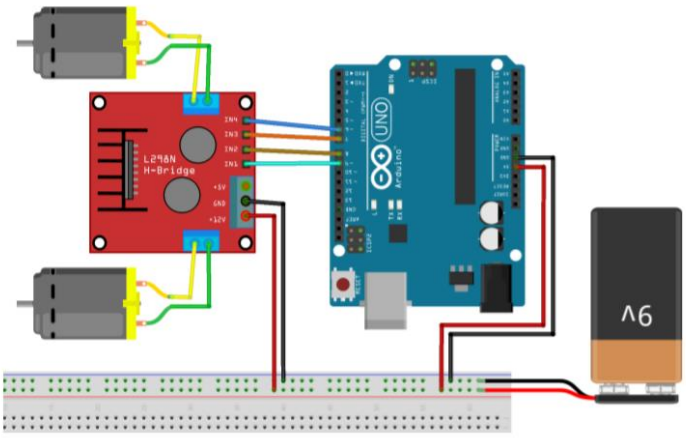
http://idboard.net:10000/reva/2018/01/07/traquer-un-objet-en-utilisant-opencv/?fbclid=IwAR1CJPUw1zk4kfeiWmQI3SDewCamKwLtFZgAZp5GOtOTiIOPpVeC39ctBt4

https://www.framboise314.fr/i-a-creez-votre-propre-modele-de-reconnaissance-dobjets-1er-partie/?fbclid=IwAR2EfJisF7UDtc2YBRWxaj_jAZBZXQjZveoKamyIk6FUNd6aZZhFdn5l7uM

12) Annexes

Annexe n°1 :

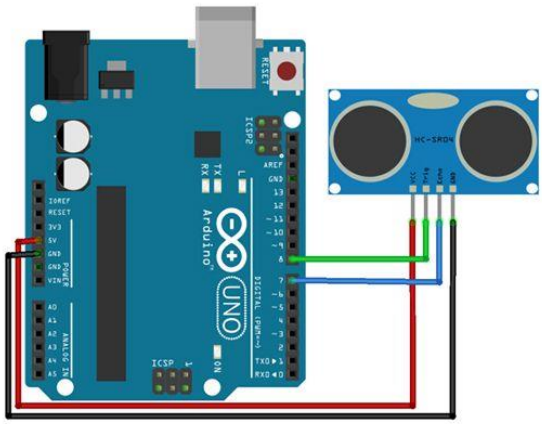
```
int motorpin1 = 3;
int motorpin2 = 4;
int pinPMoteur = 5;

void setup () {

  pinMode(motorpin1,OUTPUT);
  pinMode(motorpin2,OUTPUT);
  pinMode(pinPMoteur,OUTPUT);

}

void loop () {

  digitalWrite(motorpin1,LOW);
  digitalWrite(motorpin2,HIGH);
  analogWrite(pinPMoteur,255);
  delay(5000);
  digitalWrite(motorpin1,HIGH);
  digitalWrite(motorpin2,LOW);
  analogWrite(pinPMoteur,150);
  delay(5000);

}
```



MotoDriver 2	Arduino
Input 1	9
Input 2	8
Input 3	7
Input 4	6

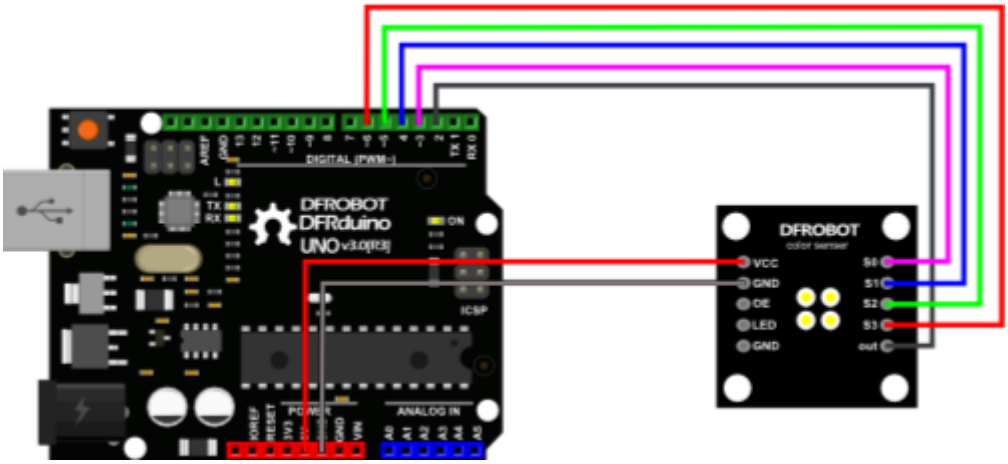
Annexe n°2 :

```
/* Utilisation du capteur Ultrason HC-SR04 */

// définition des broches utilisées
int trig = 8;
int echo = 7;
long lecture_echo;
long cm;

void setup()
{
  pinMode(trig, OUTPUT);
  digitalWrite(trig, LOW);
  pinMode(echo, INPUT);
  Serial.begin(9600);
}

void loop()
{
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  lecture_echo = pulseIn(echo, HIGH);
  cm = lecture_echo / 58;
  Serial.print("Distance en cm : ");
  Serial.println(cm);
  delay(1000);
}
```

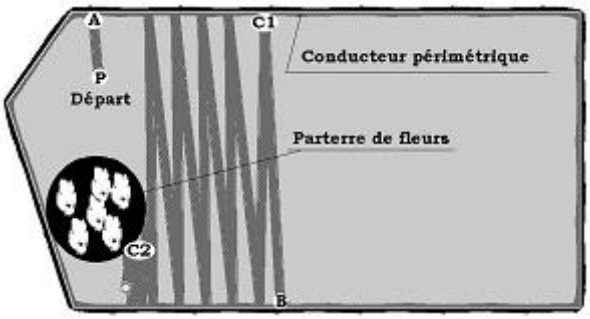


Annexe n°3 :

```
//Motor 1
const int motorPin1  = 9;
const int motorPin2  = 8;
//Motor 2
const int motorPin3  = 7;
const int motorPin4  = 6;

int speed = 180;

void setup(){

    //Set pins as outputs
    pinMode(motorPin1, OUTPUT);
    pinMode(motorPin2, OUTPUT);
    pinMode(motorPin3, OUTPUT);
    pinMode(motorPin4, OUTPUT);


    //Motor Control A in both directions
    analogWrite(motorPin1, speed);
    delay(2000);
    analogWrite(motorPin1, 0);
    delay(200);
    analogWrite(motorPin2, speed);
    delay(2000);
    analogWrite(motorPin2, 0);

    //Motor Control B in both directions
    analogWrite(motorPin3, speed);
    delay(2000);
    analogWrite(motorPin3, 0);
    delay(200);
    analogWrite(motorPin4, speed);
    delay(2000);
    analogWrite(motorPin4, 0);

}


void loop(){

}
```



Annexe n°4 :



- **T1** : Tâche d'orientation : l'opérateur ayant posé la tondeuse au sol en un point P quelconque de la parcelle, puis appuyé sur le bouton de démarrage, la tondeuse pivote autour d'un axe vertical dans le sens direct (noté Z+), et s'oriente vers le nord géographique à l'aide de la boussole électronique.

- **T2** : Tâche de recherche et de suivi de fil : la tondeuse avance ensuite vers le nord jusqu'à ce qu'elle rencontre le conducteur périmétrique (point A).
  Elle pivote dans le sens direct, et suit le conducteur périmétrique afin de tondre le pourtour de la parcelle.
  Lorsqu'elle a bouclé un tour et demi (point B), elle pivote vers l'intérieur de la parcelle et commence un cycle de tonte en "zig-zag".

- **T3** : Tâche de tonte en "zig-zag" : à chaque fois que la tondeuse rencontre le conducteur périmétrique (point C1) ou un obstacle (point C2), elle s'arrête, pivote autour d'un axe vertical d'environ 5 degrés, et repart en sens inverse.

- **T4** : Tâche de pivotement de 60 degrés: si un pivotement de 5 degrés lors de la tâche T3 l'amène à sortir du périmètre, la tondeuse pivote sur elle-même de 60 degrés et repart pour un nouveau cycle de tonte en "zig-zag".

Annexe n°5 :

```
import cv2
import numpy as np
import os
import glob
import mahotas as mt
from sklearn.svm import LinearSVC

# function to extract haralick textures from an image
def extract_features(image):
        # calculate haralick texture features for 4 types of adjacency
        textures = mt.features.haralick(image)

        # take the mean of it and return it
        ht_mean  = textures.mean(axis=0)
        return ht_mean

# load the training dataset
train_path  = "dataset/train"
train_names = os.listdir(train_path)

# empty list to hold feature vectors and train labels
train_features = []
train_labels   = []

# loop over the training dataset
print "[STATUS] Started extracting haralick textures.."
for train_name in train_names:
        cur_path = train_path + "/" + train_name
        cur_label = train_name
        i = 1

        for file in glob.glob(cur_path + "/*.jpg"):
                print "Processing Image - {} in {}".format(i, cur_label)
                # read the training image
                image = cv2.imread(file)

                # convert the image to grayscale
                gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

```

```
                # extract haralick texture from the image
                features = extract_features(gray)

                # append the feature vector and label
                train_features.append(features)
                train_labels.append(cur_label)

                # show loop update
                i += 1

# have a look at the size of our feature vector and labels
print "Training features: {}".format(np.array(train_features).shape)
print "Training labels: {}".format(np.array(train_labels).shape)

# create the classifier
print "[STATUS] Creating the classifier.."
clf_svm = LinearSVC(random_state=9)

# fit the training data and labels
print "[STATUS] Fitting data/label to model.."
clf_svm.fit(train_features, train_labels)

# loop over the test images
test_path = "dataset/test"
for file in glob.glob(test_path + "/*.jpg"):
        # read the input image
        image = cv2.imread(file)

        # convert to grayscale
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

        # extract haralick texture from the image
        features = extract_features(gray)

        # evaluate the model and predict label
        prediction = clf_svm.predict(features.reshape(1, -1))[0]

        # show the label
        cv2.putText(image, prediction, (20,30), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0,255,255), 3)
        print "Prediction - {}".format(prediction)

        # display the output image
        cv2.imshow("Test_Image", image)
        cv2.waitKey(0)
```