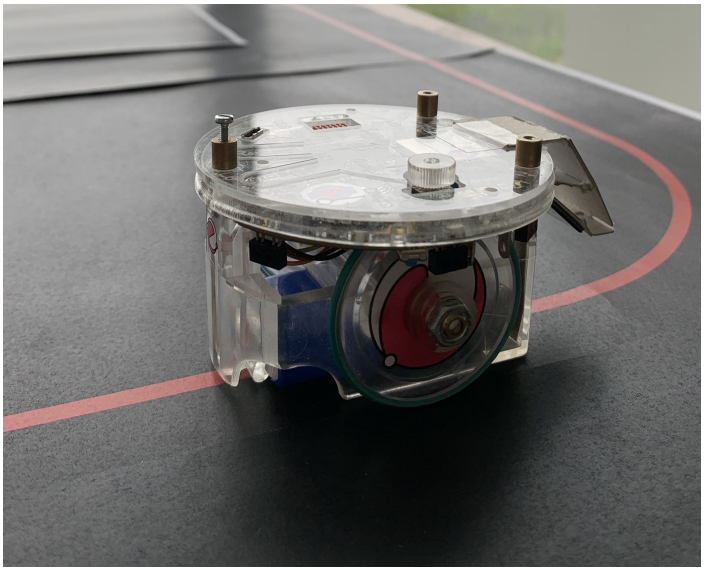
École Polytechnique Fédérale de Lausanne

MICRO 315 - Systèmes embarqués et robotique

Microtechnique BA6

Semestre de Printemps 2021

Rapport Projet E-Puck2
Vehicule de Course Autonome



Auteurs : Groupe 45
Arthur Lamour
Karim Zahra

Tables des matières

1 Introduction

Notre projet consiste à créer un véhicule se déplaçant dans un circuit et évitant des obstacles se trouvant sur celui-ci, en utilisant le robot E-Puck2 [1]. Le circuit est constitué de lignes de couleurs rouges, vertes et bleues sur un fond noir ainsi que d'obstacles de formes cylindriques et parallélépipédiques. Pour effectuer le suivi des lignes précisément, un miroir est monté devant la caméra du robot, celle-ci étant montée à l'avant, la détection d'obstacle se fait donc en sens arrière.

2 Analyse de l'environnement du robot

L'analyse de l'environnement est réalisé par l'E-Puck2 en utilisant les capteurs suivants :

- Caméra [4] : PixelPlus PO8030D CMOS, pour détecter les couleurs
- Capteurs infrarouges de distance, pour détecter les obstacles [5] : TCRT1000
- Servomoteurs [6] munis de ponts H, pour se déplacer [7]

2.1 Caméra

Pour différencier les couleurs, nous partons comme base pour le projet du travail pratique 4 [8](maintenir le robot à une distance fixe d'une ligne noire sur une feuille blanche). Chaque pixel de la caméra code la couleur sur 16 bits au format RGB565 (5 bits pour le rouge et le bleu, et 6 bits pour le vert, pour plus d'informations voir [9] et [10]), nous avons donc commencé par extraire les bits pour chaque couleur et les mettre à leurs échelles respectives (0 à 31 pour rouge et bleu, 0 à 63 pour vert).

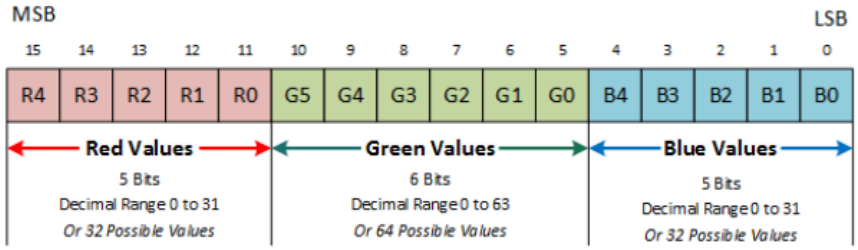


Figure 1: Représentation binaire RGB565 representation. Prise de :
`http://henrysbench.capnfatz.com/henrys-bench/arduino-adafruit-gfx-library-user-guide/arduino-16-bit-tft-rgb565-color-basics-and-selection/`

Par suite, nous utilisons des feuilles à fond noir permettant de mieux distinguer les couleurs individuellement. Cependant, le fond noir imprimmé sur les feuilles n'étant pas un corps noir idéal, un seuillage est nécessaire pour éliminer les reflets de lumière sur ce dernier (ils augmentent l'intensité relative des couleurs des pixels pouvant fausser la détection). L'éclairage de l'environnement où se trouve l'E-Puck2 est cruciale pour une bonne distinction des couleurs, s'il fait trop sombre, la couleur de la ligne se perdra dans le bruit noir, si l'espace est trop lumineux, tous les pixels seront saturés à 255 (intensité maximale des composantes RGB) rendant impossible la lecture.

Pour s'adapter à l'environnement, les seuils utilisés lors de la filtration sont réglables manuellement en utilisant le sélecteur disposant de 16 positions (position 0 : pas de seuillage, position 15 : seuillage maximal a une intensité relative de 29).

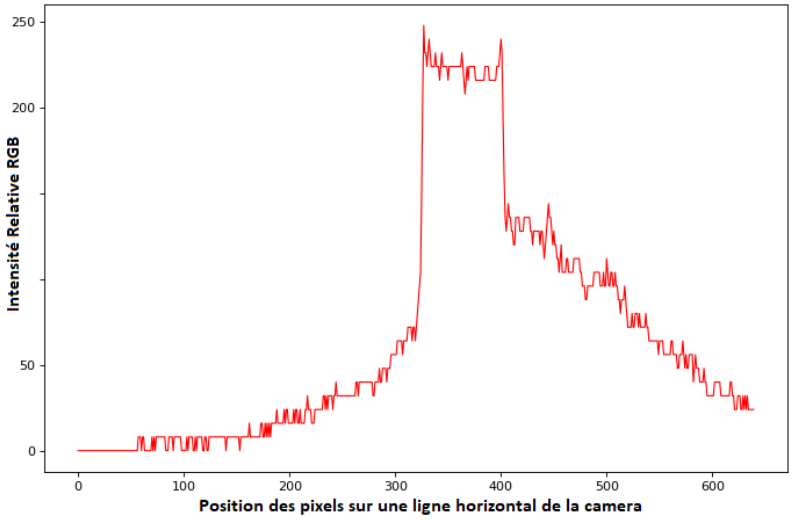


Figure 2: Détection ligne rouge sur fond noir (composante rouge non remise à l'echelle : 0 à 255)

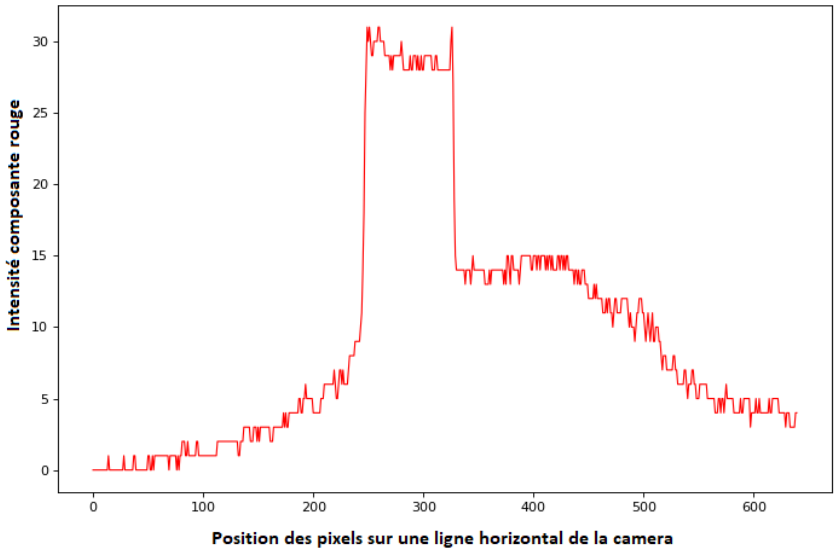


Figure 3: Détection ligne rouge avec remise à l'échelle de la composante rouge (0 à 31) sur fond noir, sans filtrage

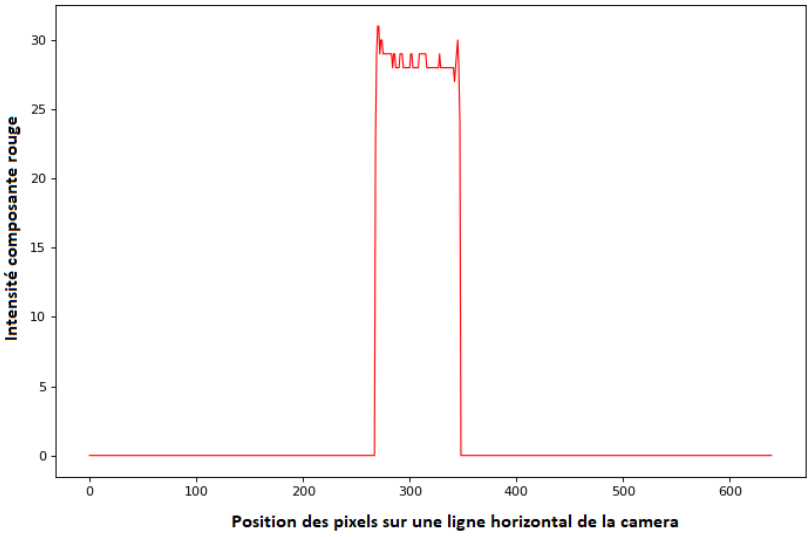


Figure 4: Détection ligne rouge sur fond noir avec remise à l'échelle et avec filtrage (utilisation d'une valeur seuile pour éliminer le bruit)

Malgré, la mise en place d'un seuillage, nous nous sommes heurtés au problème suivant : pour une ligne de couleur distincte (bleue par exemple), les composantes rouges et vertes des pixels donnent des valeurs d'intensité relativement élevée, alors que la scène était uniquement composée d'un fond noir et d'une ligne bleue.

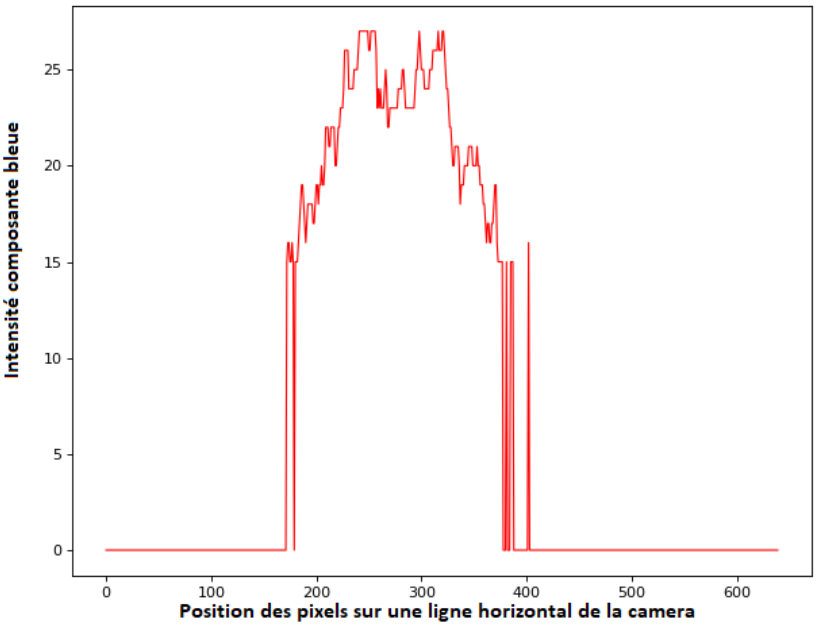


Figure 5: Intensité composante rouge pour la détection d'une ligne bleue sur fond noir avec remise à l'échelle et seuillage

De plus nous avons remarqué, que lorsque nous exposions une ligne sur un fond noir, nous obtenions des valeurs d'intensité élevées voir maximales pour cette couleur pendant un court instant, puis les valeurs d'intensité chutent drastiquement. Ceci est dû au "auto white balance" (AWB ou balance des blancs), c'est un réglage de la caméra PO8030D qui modifient les couleurs en fonction de l'éclairage. Nous supposons que l'algorithme utilisé moyenne l'intensité de l'image et réduit l'intensité

des pixels les plus colorés. L'AWB est problématique pour notre projet, car il diminue l'intensité des couleurs et rend la distinction entre couleurs plus difficile. Nous désactivons donc manuellement ce réglage et augmentons le contraste de la caméra afin d'améliorer la saturation des pixels pour pouvoir mieux distinguer les couleurs.

	Rouge	Vert	Bleu	Cyan	Jaune	Magenta
Max Pixels R	26	21	12	21	30	20
Max Pixels V	25	46	22	52	61	31
Max Pixels B	15	17	15	26	24	17

Tableau 1 : Valeurs maximales des pixels pour des lignes de différents couleurs sur fond noir (awb on)

	Rouge	Vert	Bleu	Cyan	Jaune	Magenta
Max Pixels R	31	20	14	20	30	30
Max Pixels V	22	50	22	51	59	24
Max Pixels B	19	18	22	30	22	24

Tableau 2 : Valeurs maximales des pixels pour des lignes de différents couleurs sur fond noir (awb off)

	Rouge	Vert	Bleu	Cyan	Jaune	Magenta
Max Pixels R	31	26	18	26	31	31
Max Pixels V	31	60	26	63	63	31
Max Pixels B	25	23	25	31	25	27

Tableau 3 : Valeurs maximales des pixels pour des lignes de différents couleurs sur fond noir (awb off + contraste augmenté de 64 à 80)

Note : Les cases en oranges représente une saturation.

Les lignes furent imprimées avec les imprimantes CANON de l'EPFL en codant les codes couleurs de la facon suivante : 0xFF0000 (rouge), 0x00FF00 (vert), 0x0000FF (bleu), 0x00FFFF (cyan), 0xFFFF00 (jaune) et 0xFF00FF (magenta).

Le bleu en question (0x0000FF) ne sature pas à la lumière ambiante. Cela peut être dû au fait que les imprimantes utilisent le système de couleur CMYK (cyan, magenta, jaune et noir), l'impression du bleu "pure" 0x0000FF donne un bleu plus sombre et moins intense que le bleu parfait théorique, car il est réalisé par addition des couleurs complémentaires : cyan + magenta = bleu. Une étude de couleur a été effectué avec différents bleus. Un bleu optimal fut trouvé 0x0070C0 (R = 0, G = 112 et B = 192).

	Bleu 0x0070C0
Max Pixels R	13
Max Pixels V	40
Max Pixels B	31

Tableau 4 : Valeurs maximales des pixels pour une ligne bleu 0x0070C0 sur fond noir (awb off + contraste augmenté de 64 à 80)

Similairement une étude a été réalisé pour trouver un vert optimal afin de baisser l'intensité des composantes rouges et garantir la saturation des pixels rouges pour gagner en robustesse. Le vert optimal trouvé fut 0x00E650 (R = 0, G = 230, B = 80). Il permet d'obtenir une saturation des composantes vertes et maintenir l'intensité des composantes rouges au même niveau que précédemment.

	Vert 0x00E650
Max Pixels R	26
Max Pixels V	63
Max Pixels B	25

Tableau 5 : Valeurs maximales des pixels pour une ligne bleu 0x0070C0 sur fond noir (awb off + contraste augmenté de 64 à 80)

Note : Les paramètres d'impression tels que le "Color Mangager" ont également un impacte important sur la qualité de la couleur imprimmée.

On constate que les valeurs de bleu sont très sensibles à l'éclairage, cette couleur étant plus sombre que les autres son intensité est donc plus faible. Pour bien détecter le bleu, il est primordial de travailler avec un bon éclairage, c'est-à-dire pas de rayons de lumières directes sur la feuille et pas trop sombre.

Par la suite, pour des raisons pratiques nous avons décidé de ramener le vert sur la même échelle que le bleu et le rouge à savoir 0 à 31, en gardant les 5 bits de poids fort.

2.2 Miroir

Une autre problème corps à notre projet est la fixation d'un miroir sur le robot, de sorte à ce que la caméra capture le support sur lequel il roule et n'empêche pas l'utilisation du capteur de distance "Time of Flight" et des capteurs infrarouges IR0 et IR7.

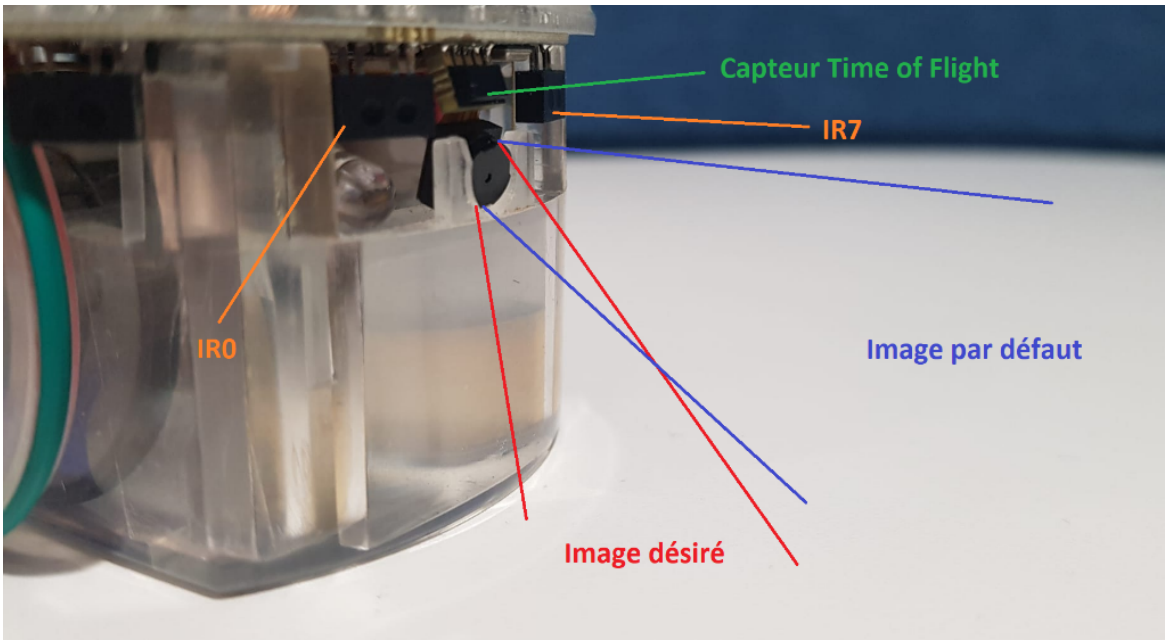
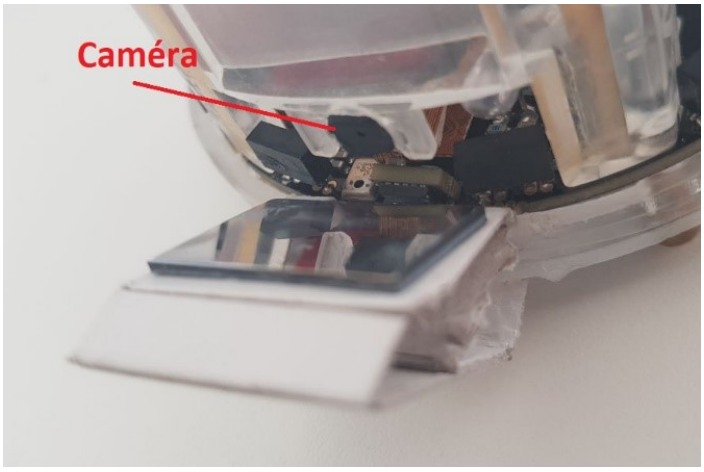


Figure 6: Considérations à prendre pour fixer le miroir

N'ayant pas d'emplacement pour fixer le miroir proche de la caméra, la solution retenue fut de coller un morceau de carton avec une forme désirée sur le robot. Cette solution a l'avantage d'être stable dans le temps : le miroir reste fixe durant le déplacement du robot, garantissant le même angle de vue à tout moment. Cependant, les capteurs Time of Flight, IR0 et IR7 ne sont plus utilisables pour notre projet, le support en carton rendant les mesures impossibles.



(a) Vue de dessous

(b) Vue de profil

Figure 7: Fixation du miroir sur l'E-Puck2

L'éclairage de la scène est très important, comme le fond noir et les lignes de couleurs sont créés par impression, une source de lumière placée au-dessus du robot pour éclairer la scène peut créer de reflets indésirables sur le noir et sur la couleur de la ligne (rouge, vert et bleu), augmentant l'intensité et pouvant mener à une saturation non désirée. Il est donc préférable d'éviter d'illuminer la scène depuis le plafond mais privilégier davantage la lumière ambiante (ex. bibliothèque RLC durant la journée). En règle générale, il faut éviter que la lumière tombe directement sur la feuille (augmentation des valeurs), mieux vaut travailler à l'ombre.

Très vite, nous nous sommes rendu compte que le contrôleur présent pour suivre une ligne rend le instable lorsqu'on se déplace vers l'arrière. Avec une seule ligne de pixels capturée, la caméra peut détecter que la ligne est bien centrée alors que le robot est mal centré sur la ligne. Il est donc impossible de suivre une ligne dans le sens inverse avec une seule image capturée.

2.3 Capteurs infrarouges

Détection d'obstacles grâce aux capteurs infrarouges. Comme vu précédemment les capteurs IR0 et IR7 étant indisponibles pour notre application, nous utiliserons les capteurs IR restant à savoir : IR2, IR3, IR4, IR5.

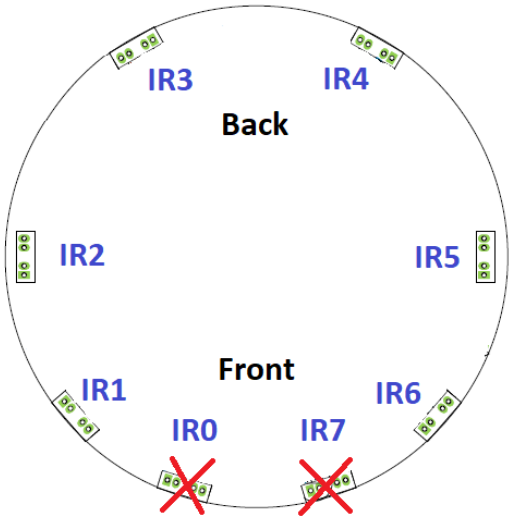
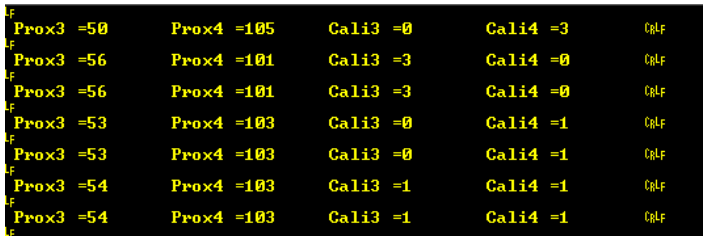


Figure 8: Disposition des capteurs IR à partir de la datasheet de l'E-Puck2 [2]

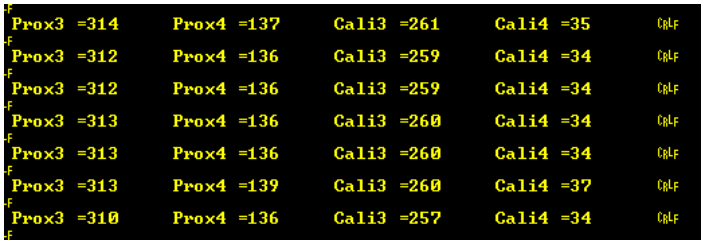
Pour pouvoir détecter les obstacles devant le robot, nous inversons l'orientation du robot, les capteurs IR3 et IR4 deviennent les capteurs avant, afin qu'un obstacle ne percute pas le montage du miroir. Ainsi, le sens de déplacement est inverse et la caméra se retrouve à l'arrière du robot. De plus nous inversons l'image avec un réglage de la caméra.

Le but étant de contourner n'importe quel obstacle de manière efficace nous avons choisi d'implémenter un contrôleur PD contrôlant la distance entre les côtés du robot et le contour de l'obstacle, une fois positionné parallèlement à l'obstacle. Ainsi nous avons choisi d'utiliser les capteurs IR2 et IR3 pour contourner un obstacle à gauche et les capteurs IR5 et IR6 pour contourner un obstacle à droite. Les valeurs utilisées pour le KP et le KD ont été déterminé manuellement à partir des essais et des cours d'automatique enseigné à l'EPFL[3].

Les capteurs infrarouges étant très sensible à la lumière ambiante nous avons choisi de les calibrer une fois au tout début. Il faut surtout faire attention d'éloigner les capteurs de tout obstacle durant cette phase de calibrage. Durant cette phase les capteurs sont mis à zéro.



(a) Pas d'obstacle



(b) Obstacle à 2.5 cm du robot

Figure 9: Captures d'écran réalisés avec le programme Realterm, des valeurs de proximités obtenus par les capteurs infrarouges IR3 et IR4. Les données brutes dans les deux premières colonnes et les données calibrés dans les deux dernières

3 Structure générale

3.1 Machine d'états finis

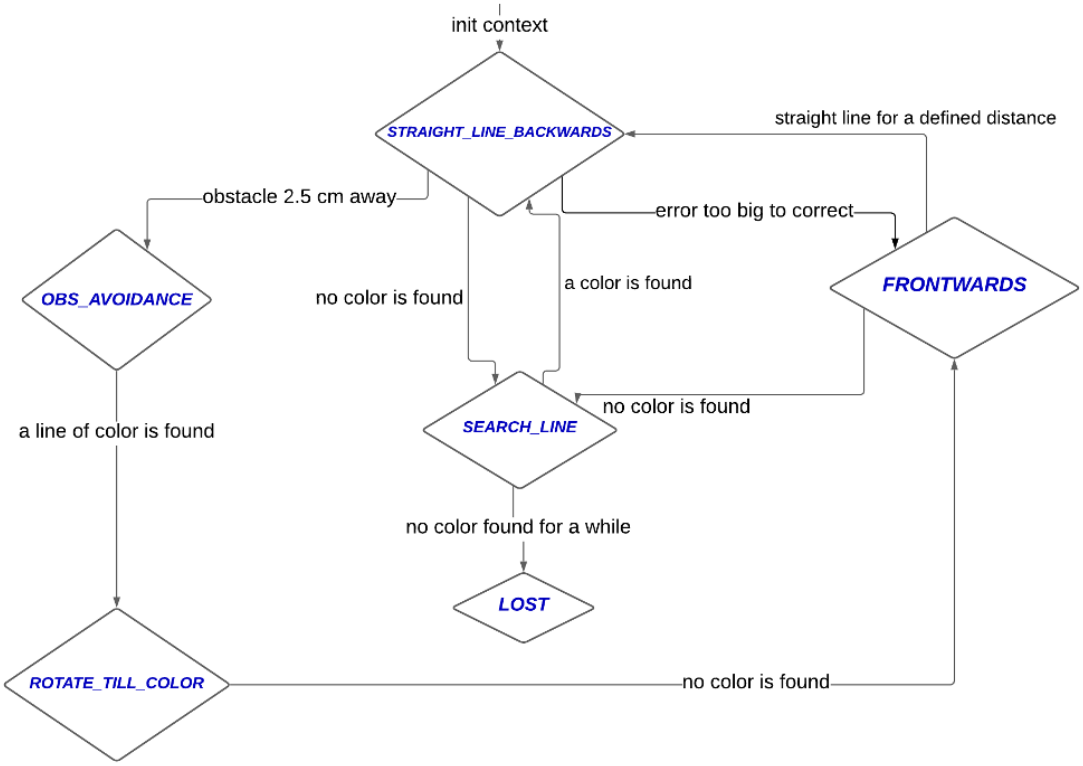


Figure 10: Diagramme de la machine d'états de mouvement

3.2 ChibiOS : les différentes threads et leurs interactions

Le programme fonctionne avec les threads suivantes :

— Moving (NORMALPRIO) : gère la machine d'états générale et les différents modes de déplacement du robot.

— CaptureImage (NORMALPRIO) : gère l'acquisition des deux images "top" et "bottom" de la caméra.

— ProcessImage (NORMALPRIO) : sert à la détection de couleur, et le milieu des lignes "top" et "bottom" de la caméra.

Ainsi qu'avec les threads de la librairie propres aux capteurs :

— pour les capteurs IR, proximity start (NORMALPRIO),

— pour l'utilisation des leds RGB, spi comm start (NORMALPRIO+1),

— pour emettre du son, playMelodyStart (NORMALPRIO)

Concernant les interactions entre les threads, nous utilisons dans nos modules deux sémaphores. En effet, ProcessImage attend sur une sémaphore image ready sem de CaptureImage que l'image soit acquise puis traite l'image.

Pour une meilleure détection de couleurs ainsi qu'une meilleure acquisition du milieu "bottom" et "top de la caméra" on suspend à des endroits critiques la thread moving et on rend volontairement

la main au scheduler. Ceci se fait par une brève attente. Ainsi on est sur que le robot se comporte correctement après un changement brusque de son environnement.

4 Conclusion

Malheureusement, malgré beaucoup d'efforts de tuning nous restons très dependant de l'éclairage du jour. Par manque d'espace et d'éclairage nous n'avons toujours pas réussi à trouver le setup idéal pour que la détection des 3 couleurs fonctionne de manière consistante sur l'entier du circuit. Comme "backup plan" on a créé un circuit uniquement en rouge. Avec uniquement une couleur, le projet fonctionne parfaitement. Le Rolex Learning Center reste toujours le meilleur endroit, tant qu'il fait beau, pour la détection des trois couleurs. Nous essaierons de créer un meilleur setup avec de la lumière artificielle au plus tot pour le jour de la présentation. Nous nous sommes rendu compte que les phases expérimentales représentaient une partie très importante dans la robotique. Ainsi que des limitations matérielles des capteurs et la gestion de leurs données malgré les perturbations.

Nous nous sommes rendu compte que les phases expérimentales représentaient une partie très importante dans la robotique, mais aussi la plus endommageante avec E-Puck2 [1]. Malheureusement, Une décharge électrique du robot a grillé un port USB-C d'un Macbook. Nous nous sommes rendu compte aussi que les limitations matérielles des capteurs et la gestion de leurs données malgré les perturbations extérieures (dues aux irrégularités de terrain et à la lumière changeante) représentait la partie la plus compliquée à gérer.

Nous sommes satisfaits du travail que nous avons pu fournir pour réaliser ce projet malgré sa difficulté et les défis dus à la crise sanitaire du covid-19. Nous avons également appris à utiliser pour la première fois l'outil Github ainsi que Github Desktop. Cela nous a fait gagner beaucoup de temps et nous a aidé pour la gestion de versions et la réalisation de tests.

Au final nous avons beaucoup aimé réaliser ce projet et espérons avoir fourni un résultat dont la qualité correspond aux attentes en fin de bachelor à l'EPFL.

References

[1] Specifications du E-Puck2 : site internet de GCtronics
https://www.gctronic.com/doc/index.php/e-puck2
Accédé le 16 mai 2021 à 20:00h.

[2] Dossier electronique de l'E-Puck2
https://moodle.epfl.ch/mod/resource/view.php?id=973661
Accédé le 16 mai 2021 à 20:00h.

[3] Prof. Alireza Karimi. Cours "Automatique et commande numerique". EPFL Lecture. Chapitre "Feedback Control Systems". 2020.
https://moodle.epfl.ch/pluginfile.php/2805924/mod_resource/content/19/Chapter4.pdf
Accédé le 16 mai 2021 à 20:00h.

[4] Camera : site internet de GCtronics
http://projects.gctronic.com/E-Puck/docs/Camera/PO8030D.pdf
Accédé le 16 mai 2021 à 20:00h.

[5] Capteurs infrarouges de distance : site internet de Vishay
https://www.vishay.com/docs/83752/tcrt1000.pdf
Accédé le 16 mai 2021 à 20:00h.

[6] Structure des roues : site internet de GCtronics
http://www.e-puck.org/index.php?option=com_content&view=article&id=7&Itemid=9
Accédé le 16 mai 2021 à 20:00h.

[7] Pont en H pour les moteurs: site internet de Allegromicro
https://www.allegromicro.com/~/media/Files/Datasheets/A3901-Datasheet.ashx?la=en&hash=6119157
Accédé le 16 mai 2021 à 20:00h.

[8] Travail pratique n°4 du cours MICRO-315 de F. Mondada
https://moodle.epfl.ch/pluginfile.php/22861/mod_resource/content/9/TP4Corrections.pdf
Accédé le 16 mai 2021 à 20:00h.

[9] Documentation sur le format RGB565 : site internet barth-dev
http://www.barth-dev.de/online/rgb565-color-picker/
Accédé le 16 mai 2021 à 20:00h.

[10] Convertisseur format RGB565 vers RGB888 de Chris Hewett : site internet de Chris Hewett
https://chrishewett.com/blog/true-rgb565-colour-picker/
Accédé le 16 mai 2021 à 20:00h.