Une image contenant objet

Description générée automatiquement

|  |  |
| --- | --- |
| KADAR Mehdi | Année 2018-2019 |

« Développement d’une Plateforme RObot MUltidisciplinaire (PROMU) »

Rapport de stage

DUT Génie Electrique et Informatique Industrielle

Du 15 avril au 21 juin 2019

INSA Lyon GE

Laboratoire Ampère

Bâtiment Saint Exupéry, 25 avenue Jean Capelle69100, Villeurbanne

|  |  |
| --- | --- |
| Tuteur entreprise : M. DELPOUX |  |
| Tuteur pédagogique : M. BIANCO | IUT Lyon 1 – Département GEII |

# Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidé lors de la rédaction de ce rapport.

Tout d'abord, j'adresse mes remerciements à mon Tuteur pédagogique, M. BIANCO de l'Université Lyon 1 qui m'a permis de postuler dans cette entreprise. Je le remercie aussi pour son écoute, ses conseils, sa confiance et les connaissances qu’il a su partager avec moi. La qualité de son encadrement en entreprise était remarquable.

Je voudrais aussi adresser toute ma gratitude à mon maître de stage M. DELPOUX, Maitre de Conférences de l’INSA Lyon, pour sa disponibilité et surtout l’autonomie qu’il m’a offert pendant ce stage. Grâce aussi à sa confiance j'ai pu m'accomplir totalement dans mes missions. Il fut d'une aide précieuse dans les moments les plus délicats.

Il est aussi important que je site mon binôme M. VALETTE, qui a travaillé avec moi tout au long du stage et qui m’a permis de passer un bon moment durant ces 10 semaines.

Je remercie également M. BRUN, M. GRENIER ainsi que tout le personnel pour leur accueil, leur conseille ainsi que leur bienveillance.

Enfin, je tiens à remercier toutes les personnes qui m'ont conseillé et relu lors de la rédaction de ce rapport de stage.

Table des matières

[Remerciements 3](#_Toc11597053)

[Table des illustrations 5](#_Toc11597054)

[Introduction générale 5](#_Toc11597055)

[Contexte de la mission 6](#_Toc11597056)

[1) Le groupe INSA 6](#_Toc11597057)

[2) INSA Lyon 7](#_Toc11597058)

[3) Département Génie Electrique 8](#_Toc11597059)

[Présentation technique de la mission 9](#_Toc11597060)

[1) Objectif(s) 9](#_Toc11597061)

[2) Analyse de l’existant 10](#_Toc11597062)

[a) Les robots mobiles 10](#_Toc11597063)

[b) Le bras robotique 11](#_Toc11597064)

[3) Les contraintes et enjeux 12](#_Toc11597065)

[Mise en œuvre de la mission 12](#_Toc11597066)

[1) Découverte de la mission 12](#_Toc11597067)

[2) Mise en place de la liaison série 13](#_Toc11597068)

[3) Contrôle des moteurs 16](#_Toc11597069)

[4) Etape 4 : Mise en place de la collaboration entre le robot mobile et le bras 17](#_Toc11597070)

[5) Asservissement du robot mobile 17](#_Toc11597071)

[6) Mise en place du projet buildroot 17](#_Toc11597072)

[7) Synchronisation entre le robot mobile et le bras électronique 17](#_Toc11597073)

[Bilan de la mission (intermédiaire ou final) 18](#_Toc11597074)

[Résultats et évaluation 18](#_Toc11597075)

[Difficultés rencontrées et analyse des solutions retenues 18](#_Toc11597076)

[Perspectives d’avenir 18](#_Toc11597077)

[Bilan personnel 18](#_Toc11597078)

[ANNEXES 18](#_Toc11597079)

[Annexes 1 : Mise en place de la liaison série sur la Raspberry 18](#_Toc11597080)

[Annexes 2: Ficher SerialObj.h 23](#_Toc11597081)

[Annexes 3: Ficher SerialObj.cpp 25](#_Toc11597082)

# Table des illustrations

[Figure 1: Logo Group INSA 7](file:///C:\Users\mehdi\Documents\ancien%20bureau\projet_arduino\Projet_1\KADAR_INSA.docx#_Toc11597176)

[Figure 2: Carte montrant la localisation de l'INSA de Lyon 8](file:///C:\Users\mehdi\Documents\ancien%20bureau\projet_arduino\Projet_1\KADAR_INSA.docx#_Toc11597177)

[Figure 3: Schéma du système à développer 10](file:///C:\Users\mehdi\Documents\ancien%20bureau\projet_arduino\Projet_1\KADAR_INSA.docx#_Toc11597178)

[Figure 4: Image des capteurs placé sous le robot 10](file:///C:\Users\mehdi\Documents\ancien%20bureau\projet_arduino\Projet_1\KADAR_INSA.docx#_Toc11597179)

[Figure 5: Suivie de la ligne à l'aide de la caméra 13](file:///C:\Users\mehdi\Documents\ancien%20bureau\projet_arduino\Projet_1\KADAR_INSA.docx#_Toc11597180)

[Figure 6: Synoptique du Robot mobile 14](file:///C:\Users\mehdi\Documents\ancien%20bureau\projet_arduino\Projet_1\KADAR_INSA.docx#_Toc11597181)

[Figure 7: Organigramme du programme de teste de la liaison série 16](file:///C:\Users\mehdi\Documents\ancien%20bureau\projet_arduino\Projet_1\KADAR_INSA.docx#_Toc11597182)

[Figure 8: Organigramme de l'application qui permet de contrôler les roues à l'aide de la liaison série 17](file:///C:\Users\mehdi\Documents\ancien%20bureau\projet_arduino\Projet_1\KADAR_INSA.docx#_Toc11597183)

[Tableau 1: Les différents formations de l'INSA de LYON 8](file:///C:\Users\mehdi\Documents\ancien%20bureau\projet_arduino\Projet_1\KADAR_INSA.docx#_Toc11597410)

# Introduction générale

Du 15 avril au 21 juin 2019, j’ai effectué mon stage de fin de DUT en Génie Electrique et Informatique Industrielle (GEII) au sein de l’INSA Lyon située à Villeurbanne. Au cours de ce stage dans le département Génie Electrique, j’ai pu m’intéresser aux stratégies de pilotage et à la collaboration de système électronique complexe.

L'INSA Lyon est la première école d'ingénieurs postbac de France. Elle accueille chaque année une grande diversité de profils parmi les meilleurs bacheliers de France. Sa notoriété en France, en Europe et à l’international et dû notamment à sa proximité avec les entreprises mais essentiellement grâce à l’excellence de son enseignement.

Aujourd’hui, elle développe une nouvelle plateforme pédagogique pour leurs futurs étudiants. A ce jour, cette plateforme comporte des robots mobiles pilotés pour suivre une ligne. Récemment des bras montés sur un rail viennent compléter la plateforme. Mon projet a pour but de mettre en œuvre, dans un premier temps, des stratégies de pilotage pour chaque système puis, dans un second temps, de faire collaborer les bras manipulateurs avec les robots mobiles.

Dans ce rapport, je commencerai par vous présenter l’organisme qui m’a accueillie. Par la suite, je parlerai du projet que l’on m’a confié en partant de l’étude, en passant par la conception et en finissant par les différents tests effectués. Pour finir, je développerai cette expérience d’un point de vue plus personnel pour décrire ce que m’ont apporté ce stage et ces deux années au sein de l’IUT (Institut Universitaire Technologique).

# Contexte de la mission

## Le groupe INSA

Le groupe INSA (INSA) est constitué de 14 établissements publics français de recherche et d’enseignement supérieur. Plus grand ensemble de formation d'ingénieurs en France (10 % des ingénieurs), ils délivrent principalement le diplôme d’ingénieur après une formation de cinq années après le baccalauréat. Leurs étudiants sont appelés Insaïens ou parfois Insaliens.

Il est constitué des INSA fondatrices qui ont une identité commune et des écoles partenaires qui sont historiquement indépendantes. L'ensemble recrute grâce au concours commun du groupe INSA (pour l'admission postbac seulement).

L'histoire de ce groupe entre dans le cadre d'une politique de décentralisation. De création assez récente, les INSA ont essentiellement un objectif social qui est de donner accès au plus grand nombre d'étudiants à des études supérieures de qualité quel que soit leur milieu d'origine.

Figure 1: Logo Group INSA

## INSA Lyon

Créé en 1957 par le philosophe Gaston Berger et le recteur Jean Capelle, l’INSA Lyon est le plus ancien et le plus important des six établissements du Groupe INSA.

L’INSA Lyon compte treize unités de recherche en rattachement principal, et sept en rattachement secondaire ; ces unités sont pour la plupart des UMR avec le CNRS. L’INSA Lyon est l’établissement support de deux écoles doctorales, est cohabilitée pour six autres, et est associée à une dernière.

INSAVALOR, filiale de l’INSA Lyon, a pour mission principale le transfert et la valorisation des activités de recherche menées par les 23 laboratoires de recherche.

Le directeur de l’INSA de Lyon est M. Maurincomme depuis 2011. C’est un ingénieur français, et ex-vice-président du Marketing & Business Développent chez Agfa Healthcare (Belgique).

En 2018, l’INSA de Lyon comportait 5383 étudiants avec un effectif de 1960. Leur budget c’était élevé à 55 600 000 euros.

Une image contenant texte, carte

Description générée automatiquementComme le montre la carte dans la figure 2, leurs locaux se situe à Villeurbanne dans le campus de la Doua.

Figure 2: Carte montrant la localisation de l'INSA de Lyon

Tableau 1: Les différents formations de l'INSA de LYON

Une image contenant capture d’écran

Description générée automatiquement

## Département Génie Electrique

Le département de Génie Electrique assure une formation d’ingénieurs pluridisciplinaires dans le domaine des systèmes électriques, permettant de s'insérer activement dans tous les secteurs du génie électrique : la conversion d’énergie, les systèmes embarqués, les télécommunications, l'automatisme industriel, le traitement du signal et des images, les véhicules électriques, les réseaux de distribution de l'énergie.

L’objectif de la formation est l’acquisition de connaissances théoriques et de compétences pratiques dans les domaines de l’Electronique, de l’Electrotechnique, de l’Automatique, de l’Informatique Industrielle et des Télécommunications, cela se formalise par 55% des enseignements du département. Les 45% restant sont à déterminer par l'étudiant afin de développer son projet professionnel : l’international, le stage industriel (6 mois), le Projet de Fin d’Etude (5 mois), l'option de 5e année.

# Présentation technique de la mission

## Objectif(s)

L’objectif de mon stage est de mettre en place une plateforme robot multidisciplinaire qui servira de base pour la création de nouveau travaux pratique pour les futurs étudiants de l’INSA.

Des robots mobiles viendront se déplacer le long d’un parcours délimité par une ligne noire. Autour de la ligne, deux bras électroniques serons présents. Les robots devront s’arrêter à coté de chacun d’eux.

Quand le bras manipulateur 1, représenté dans la figure 2, détectera l’arrivé d’un robot, il récupèrera un cube dans un des stocks placés à côté de lui et viendra charger ce cube sur le robot. Quant au bras manipulateur 2, il s’occupera de récupérer le cube présent sur le robot et de le décharger sur une des boîtes présente le long du rail.

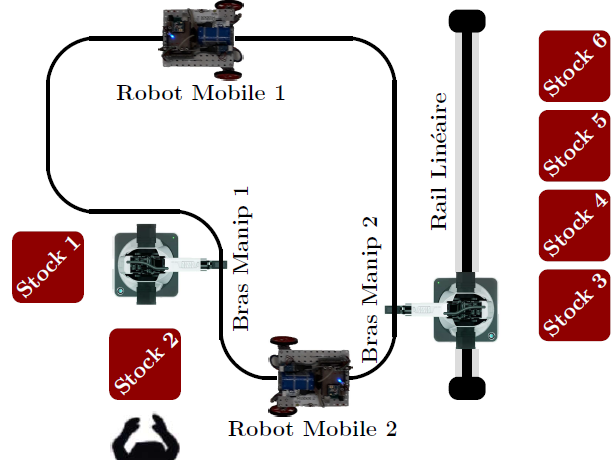


Figure 3: Schéma du système à développer

## Analyse de l’existant

### Les robots mobiles

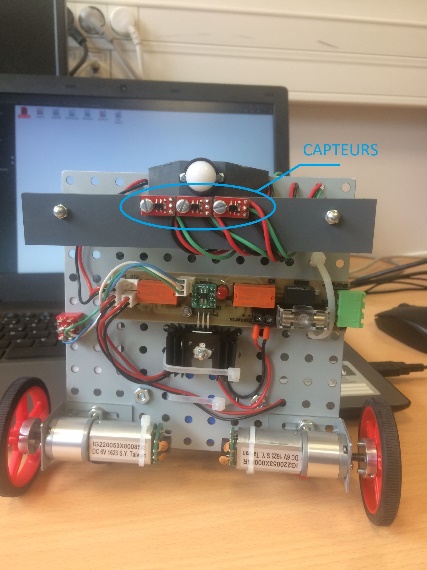
Lors de mon arrivé, les robots mobiles ont déjà été programmé dans le but de suivre la ligne. C’était le sujet d’un des TP d’automatisme élaboré par les enseignants de l’INSA. La détection de la ligne se faisait à l’aide de capteurs placé sous le robot (voir figure 4).

Figure 4: Image des capteurs placé sous le robot

Le principe de fonctionnement est simple. On envoie une commande constante aux roues et en fonction de si la ligne est détectée par le capteur de droit, du milieu ou de gauche, on ajoute une constant avec un coefficient proportionnel sur une des deux roues afin de recentrer le robot sur la ligne.

Cependant, l’acquisition des données à l’aide de ces capteurs présentait une limite au niveau de la performance du suivie de ligne. En effet, on pouvait constater des oscillations assez fréquentes, ce qui pourrais, par la suite, dans le cas d’un développement future, s’avéré gênant ou pourrais entraver le fonctionnement global du système.

L’acquisition des données se fera donc à l’aide d’un autre moyen plus efficace mis en place durant ce stage.

Mon maître de stage m’a donc confié la mission de mettre en place un asservissement visuel à l’aide d’une caméra brancher à un Raspberry. Cette méthode devrait pouvoir améliorer l’efficacité du robot.

Pour être claire, le robot sera divisé en deux parties. La première est l’Arduino qui s’occupe de contrôler les roues et la deuxième est la Raspberry qui s’occupe de détecter la ligne.

Afin de pouvoir commander les roues, l’Arduino aura besoin des informations collecté par la caméra de la Raspberry. C’est pourquoi j’ai dû mettre en place un Protocol de communication entre l’Arduino et la Raspberry afin qu’ils se comprennent.

### Le bras robotique

Le bras robotique a été développé par la société Yuejiang et lui ont attribué le nom de Dobot Magician. Ce produit a été imaginé dans le but de développer les compétences en programmation des étudiant tout en leur proposant un outil polyvalent et intelligent pour de nombreuses opérations de robotique industrielle.

Cet outil nous sera très utile durant ce stage, non seulement afin de réaliser le système que je vous décris un peu plus tôt mais aussi car elle est parfaitement adaptée au visé pédagogique que nous essayons d’atteindre.

Ce bras robotique se programme normalement à l’aide de Dobot Studio, l’IDE développer par son créateur. Cependant, pour ce qui concerne le control du bras, notre tuteur de stage à voulu que nous nous servions du protocole de communication qui est fournie avec le Dobot afin de la contrôler avec la liaison série de l’Arduino.

### Les contraintes et enjeux

Comme mon projet doit servir de base pour la création de nouveau travaux pratique pour les futurs étudiants de l’INSA, il a fallu que j’axe le développement de mon code dans ce sens-là.

Mon code doit être robuste et abordable pour des étudiants de l’INSA et doit également garder une certaine flexibilité afin d’être utilisé de plusieurs manière différente.

Aussi comme nous utilisons des Arduino, on a dû apprendre le développement en c++, un langage qui n’a pas été enseigner lors de notre DUT.

La Raspberry que j’ai utilisé le système d’exploitation Raspbian. Je devais donc me familiariser avec l’environnement Linux.

Enfin, la faite de travailler en collaboration avec d'autres professeurs nous a obligé à utiliser un outil collaboratif afin de partager nos avancements. C’est pourquoi un dépôt git a été créé. On devait donc apprend à utiliser cet outil qui est d’ailleurs très utilisé chez les développeurs en informatique.

# Mise en œuvre de la mission

## Découverte de la mission

La première semaine de stage m’a permis de me familiariser avec le matériel et l’environnement de travail. Durant cette période j’ai pu mettre aux claires les tâches que je devais accomplir avec leur ordre de priorité.

Comme nous étions en binôme, nous devions nous répartir les taches de manière à avancer plus vite. Après une analyse du cahier des charge nous avons pu diviser ce projet en trois gros blocks.

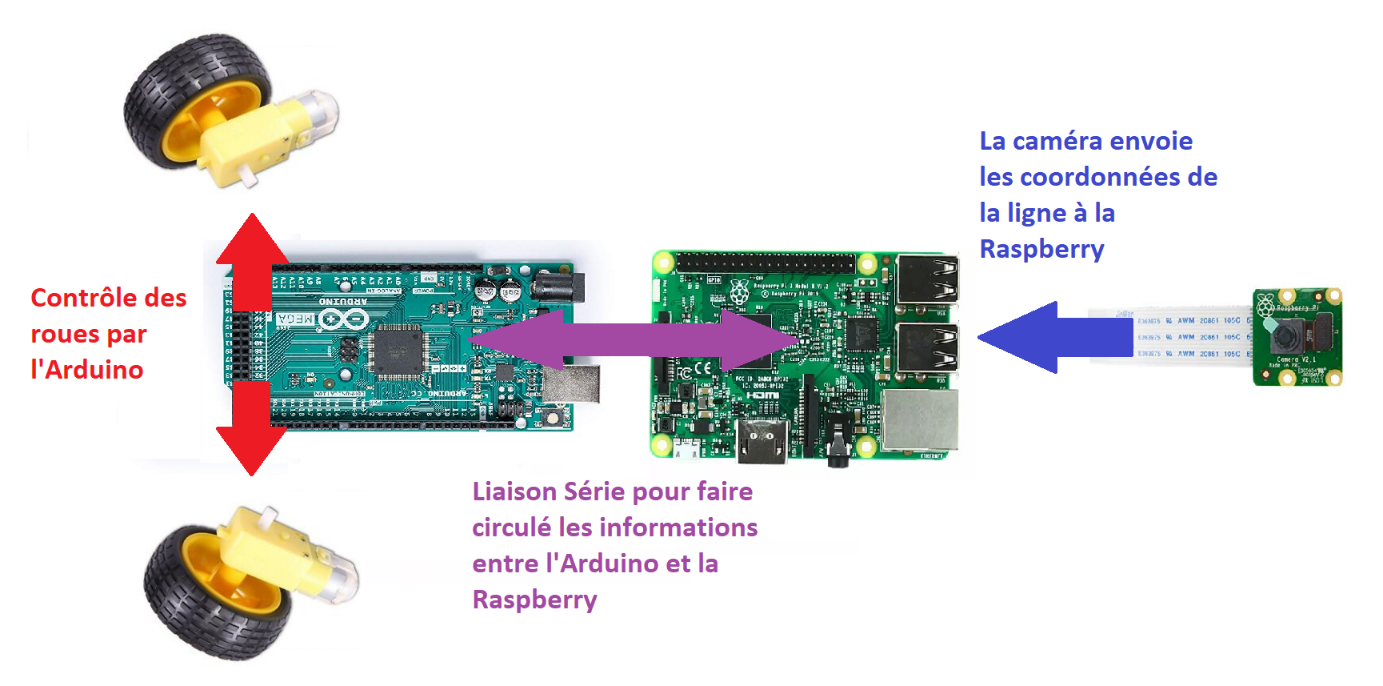
La première partie consiste à gérer le suivit de ligne du robot à l’aide de la caméra branché sur la Raspberry. Voici un schéma qui explique le système à obtenir :

Figure 5: Suivie de la ligne à l'aide de la caméra

La deuxième partie concerne le bras robotique. Le protocole de communication entre l’Arduino et le Dobot Magician doit être respecté. Aussi, des fonctions devrons être développer afin d’avoir accès à toutes les fonctionnalités que peu offrir cet appareil. Le code doit être assez claire et compréhensible pour pouvoir être réutilisable par une tierce personne.

Parmi toutes les commandes qui peuvent être développé pour contrôler le bras, certaine sont plus utile pour notre application que d’autre. On a donc dû faire une sélection en se basant sur ce dont on aura besoin pour notre système final.

La dernière partie est plus complexe. On devra trouver un moyen qui nous permettra de synchroniser le robot avec le bras. Cette partie ne peut être abordé que si les deux premières parties sont fini.

## Mise en place de la liaison série

Consternant la répartition des tâches, je me suis occupé de développer la première partie : le robot mobile.

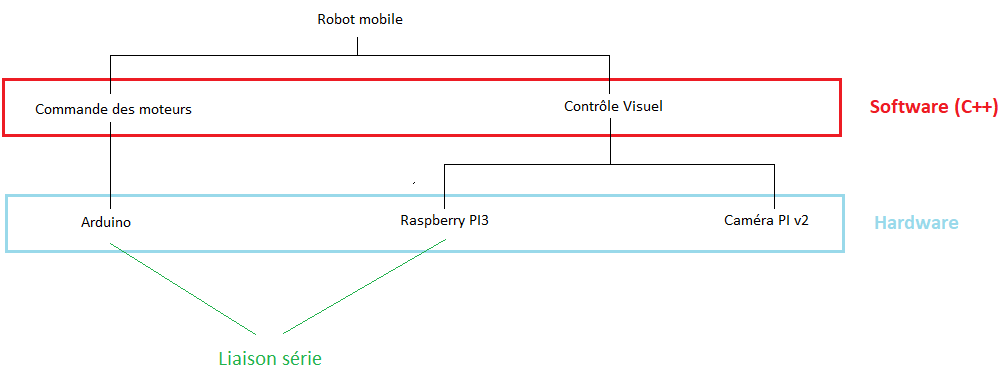


Figure 6: Synoptique du Robot mobile

Pour commencer, j’ai fait un synoptique qui m’a servie de structure de base afin d’avancer efficacement dans mon travail.

On m’a mis à disposition une Arduino Méga 2560 et une Raspberry pi3 b+ afin de réaliser mes tests.

En ce qui concerne l’environnement de développement, j’ai travaillé sur un poste fixe Windows avec l’IDE Arduino. Du côté de la Raspberry j’ai pu installer Eclipse.

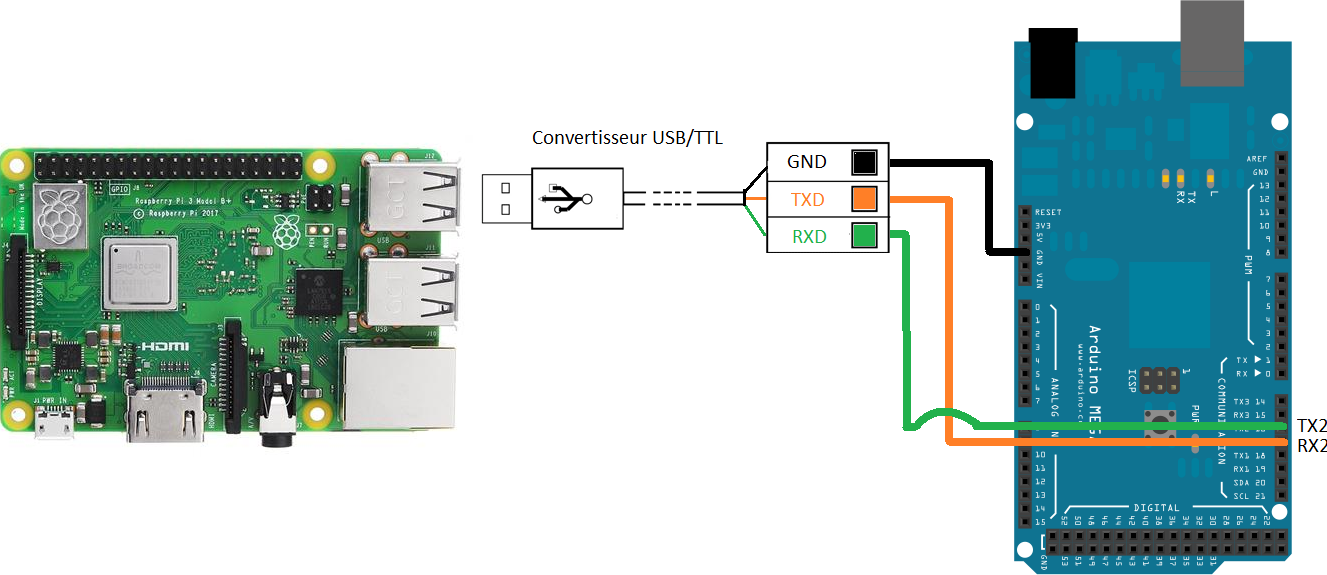
Afin de connecter l’Arduino à la Raspberry, j’ai utilisé, comme vous pouvez le voir figure 9, un convertisseur USB/TTL.

Figure 9 : schéma de la liaison physique entre la Raspberry et l'Arduino

Comme j’ai déjà eu l’occasion de travailler avec des Arduino, j’ai pu facilement trouver comment envoyer des octets dans la liaison série. Il suffit d’utiliser les méthodes de la classe Serial qui est intégrer dans la bibliothèque de base de l’Arduino.

Cependant, c’était ma première expérience avec une Raspberry. Il a donc fallu que je trouve comment activer le port série.

La configuration de la Raspberry afin d’activer la liaison série m’a pris du temps. Je vous laisse lire en annexe 1 tous les étapes de configuration afin d’arriver à ce but. J’ai dû rester la plus par du temps sur des forums en anglais.

Ensuite, une fois le port série de la Raspberry activé, il fallait que je sache comment envoyer des données dans la liaison série en passant par du code en C/C++. C’est là que les cours en Linux embarqué mon été le plus utile. J’ai pu y apprendre comment on accède aux ports séries sous Linux, c'est-à-dire comment y lire des caractères et en envoyer, comment modifier les paramètres de la liaison, etc...

Sous les systèmes de type UNIX comme Linux, tout (ou presque) est vu comme un fichier. Les périphériques sont stockés dans le ficher /dev. En connectant le convertisseur USB/TTL, figure 9, à la Raspberry, un nouveau ficher se crée ce le dossier avec le nom de « ttyUSB0 ». Afin d’accéder à la liaison série on devra donc de connecter au ficher : /dev/ttyUSB0.

En plus des compétences de base que j’ai pu acquérir, j’ai dû faire quelque recherche supplémentaire sur le web pour optimiser mon code et débugger certain problème.

Un peu plus tard, j’ai pu récupérer une bibliothèque gérant la liaison série en C écrite par M. Bianco, mon professeur de linux embarqué et tuteur pédagogique. J’ai pu développer ma première application. Voici son organigramme :

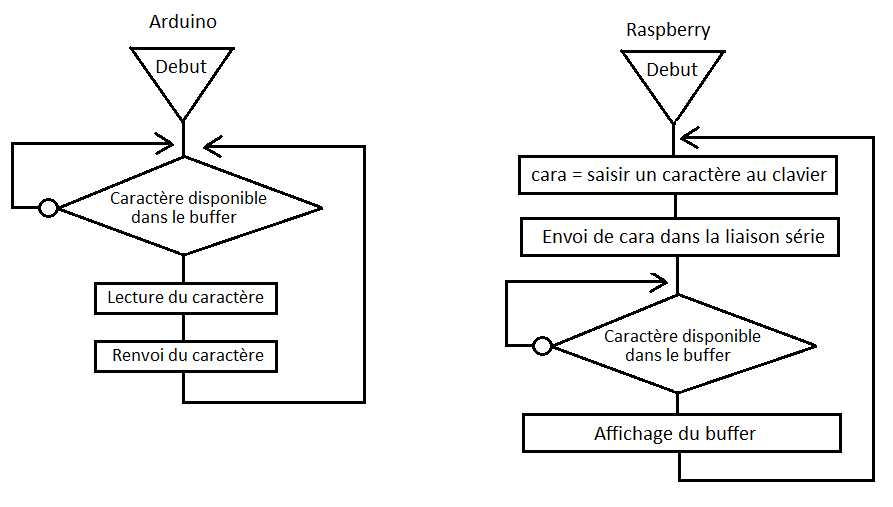


Figure 7: Organigramme du programme de teste de la liaison série

Le programme implanté dans l’Arduino est simple. Il fait un écho de ce qu’il reçoit dans sa liaison série. Quant à la Raspberry, l’utilisateur envoie un caractère dans la liaison série et attend un retour.

Une fois mon application fonctionnelle, il fallait que je change le code le la Raspberry. En effet, mon tuteur voulait qu’on code en c++. J’ai donc créé la classe « SerialObj » en me basant sur le code de M. Bianco. Vous pouvez le consulter en annexe 2 et 3. Ensuit, il à fallu que je teste cette classe avec la même application.

## Contrôle des moteurs

Maintenant que la liaison série est prête, je peux passer à l’étape suivante. Il faut que je réussie à contrôler les moteurs à l’aide de l’Arduino. Pour cela, j’ai commencé étudier le code qui à été développé par M. Delpoux et M. Bianco qui permettait au robot de suivre la ligne avec les capteurs de luminosité. J’ai récupéré le code qui peut m’être utile, c’est-à-dire la classe « Moteur ». Parmi les attributs de cette classe, le plus utile est celle qui me permet d’envoyer une commande au moteur. J’ai pu donc réaliser une seconde application. Vous pouvez retrouver son organigramme ci-dessous figure 8.

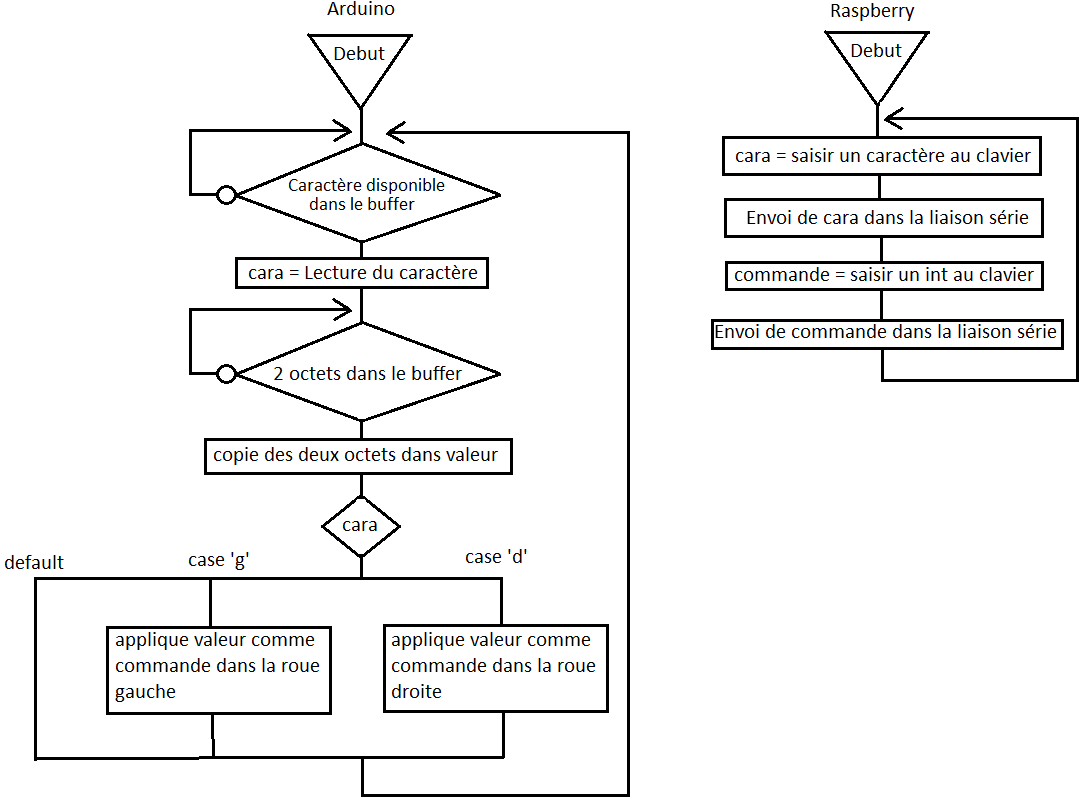


Figure 8: Organigramme de l'application qui permet de contrôler les roues à l'aide de la liaison série

Cette application me permet de piloter les roues à l’aide de la Raspberry en envoyant des commandes à l’Arduino.

La liaison série fonctionne, le contrôle des roues par la Raspberry aussi. Il ne me reste maintenant que deux étapes. L’acquisition et le traitement des données de la caméra.

## Acquisition des données de la caméra

Pour l’acquisition des donné de la caméra, on a dû utiliser OpenCV (Open Source Computer Vision). C’est une bibliothèque proposant un ensemble de plus de 2500 algorithmes de vision par ordinateur, accessibles au travers d'API pour les langages C, C++, et Python. Elle est distribuée sous une licence BSD (libre) pour les plateformes Windows, GNU/Linux, Android et MacOS.

Afin de gagner du temps, le code de l’acquisition des données via la caméra a été développé par M. Grenier, une enseignant de l’INSA qui s’occupe également du projet, pendant que je travaillais sur la liaison série. J’ai mis en annexe 4 la fonction « process() » qui s’occupe d’afficher les coordonnées de la ligne capté par la caméra.

Cependant, la version d’OpenCV utilisé n’était pas compatible avec la version de Raspbian de notre Raspberry. On pouvait exécuter son code sur un pc Linux mais pas sur la Raspberry. Afin de palier à ce problème, on à utilisé Buildroot afin de générer une image Linux pouvant être transporté dans la carte SD. C’est M. Bianco, mon tuteur pédagogique, qui s’est occupé de le configuré.

Maintenant le code est exécutable sur la Raspberry pi.

## Traitement des données de la caméra

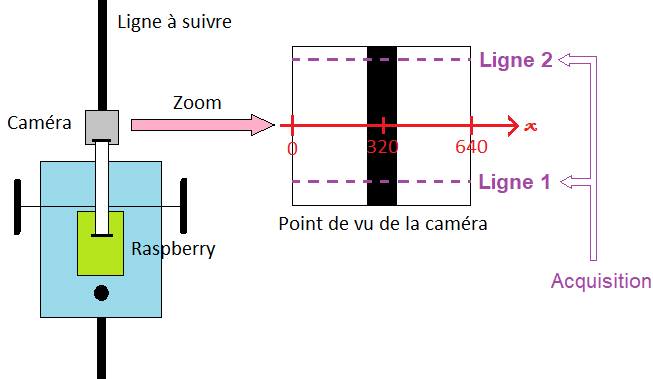
Maintenant qu’on a un système capable de réaliser l’acquisition des données, on peut passer au traitement. Afin que le robot suive la ligne, il faut réaliser un asservissement dont la consigne dépendra de la position du robot.

Figure 9: Modélisation de l'acquisition des données de la caméra

Le code de détection de ligne nous renvoie la position de la ligne selon un axe x allant de 0 à 640. J’ai donc commencé par changer l’origine afin d’avoir des valeurs comprises dans l’intervalle [-320 ;320]. Ci-dessous, figure 10, vous pouvez retrouver le code qui m’a permis de changer l’origine.

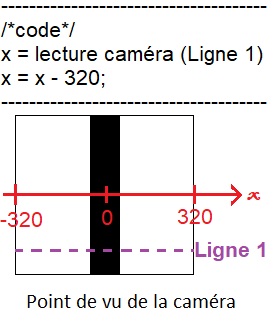
Le code de M. Grenier fait le calcule du barycentre. Ce qui permet de garder les coordonnées du centre de la ligne noire.

Figure 10: Changement de l'origine

Le but de l’asservissement est de ramener X à 0.

Premièrement, on envoie une commande constante au deux roue (même valeur) afin que le robot avance tout droit. Ensuit en regarde la position x de la ligne à l’aide de la caméra. Si x<0, c’est que la ligne se trouve à gauche. On augmentera donc la commande de la roue gauche et on diminuera celle de la roue droite afin de faire tourner le robot vers la droite.

Vous pouvez voir le schéma bloc du système ci-dessous (figure 9) :

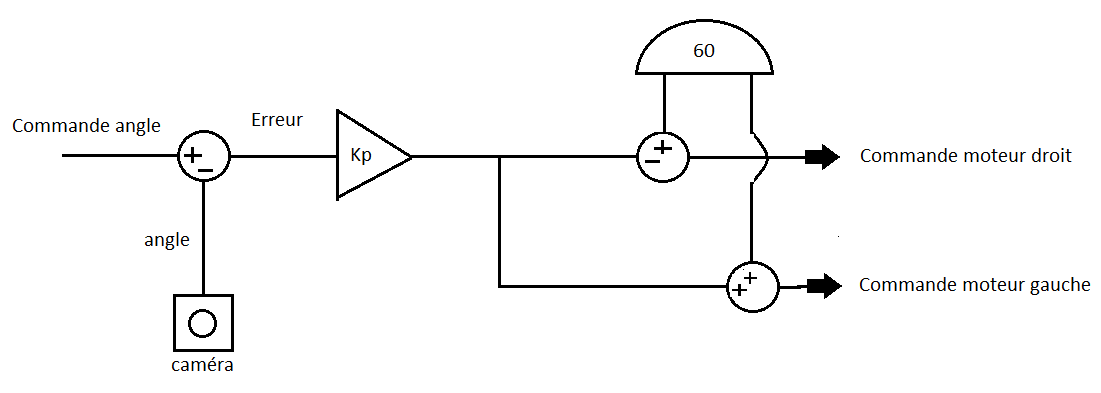


Figure 11: schéma bloc du système

Commande moteur droit (ComD)= 60 + Kp\*(Commande angle – angle).

Commande moteur gauche (ComG)= 60 + Kp\*(Commande angle – angle).

La valeur de « angle » envoyé par la caméra est comprise entre -320 et +320 (comme nous avons changé l’origine afin de simplifier les calculs)

Puisqu’on veut que le robot soit centré on prendra Commande angle = 0.

Finalement on aura :

ComD = 60 – Kp\*angle

ComG = 60 + Kp\*angle

Si l’angle est positif (donc position de la ligne à droite de la caméra), la roue gauche tournera plus vite que la roue droite, le robot ira à droite. Inversement, si l’angle est négatif (donc position de la ligne à gauche de la caméra), la roue gauche tournera moins vite que la roue droite, le robot ira à gauche.

## Mise en place du projet buildroot

## Synchronisation entre le robot mobile et le bras électronique

# Bilan de la mission (intermédiaire ou final)

## Résultats et évaluation

## Difficultés rencontrées et analyse des solutions retenues

## Perspectives d’avenir

# Bilan personnel

# ANNEXES

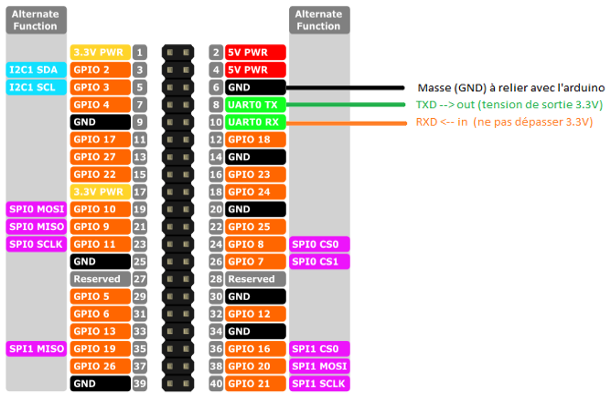
## Annexes 1 : Mise en place de la liaison série sur la Raspberry

**Hypothèse de départ:**

-Raspbian est déja Installez dans une carte SD.

-La Raspberry boot correctement sur la carte SD

-On veut pouvoir communiquer en liaison serie en utilisant les ports GPIO de la Raspberry.

Identification des pins UART de la Raspberry

+Info : <https://docs.microsoft.com/en-us/windows/iot-core/learn-about-hardware/pinmappings/pinmappingsrpi>

Activation du port série

|  |  |
| --- | --- |
| **Etape 1** | Connectez vous en terminal (en mode texte ou ouvrez un terminal sur le bureau)  Démarrez l’utilitaire de configuration système avec :  ---->sudo raspi-config  Une interface graphique s'affiche à l'écran ! |
| **Etape 2** | Sélectionnez “**5 options d’interface**“. |
| **Etape 3** | Sélectionner “**P6** Série”. |
| **Etape 4** | Un écran vous demande si vous souhaitez qu’un écran de login soit accessible via le port série.  On veut utiliser le port série pour contrôler d’autres périphériques série (Arduino), on sélectionne donc **Non**. |
| **Etape 5** | Un autre écran nous demande si vous souhaitez que le matériel du port série soit activé  On sélectionne **Oui**. |
| **Etape 6** | Sur l’écran principal de configuration de **raspi-config**, On sélectionne “**Terminer**” et redémarrez votre Raspberry Pi. |

Le port série mini UART doit maintenant être activé !!

|  |  |
| --- | --- |
| **Etape 7** | Verification  ---> ls /dev/serial0 -l  On doit obtenir ça :    ttyAMA0 et serial0 permettent d'accéder à la même liaison.  On utilisera de preferance **serial0**. |

Réglage du port série

|  |  |
| --- | --- |
| **Etape 1** | Connectez-vous en terminal (en mode texte ou ouvrez un terminal sur le bureau) |
| **Etape 2** | On modifie le fichier config.txt  ---> sudo nano /boot/config.txt  Ajouter à la fin du fichier :  **pi3-miniuart-bt**  et  dtoverlay=pi3-miniuart-bt  Quittez l’éditeur en sauvegardant vos modifications (CTRL X) |
| **Etape 3** | Redémarrer  ----> sudo reboot |

Site utilisé :

<https://www.framboise314.fr/utiliser-le-port-serie-du-raspberry-pi-3-et-du-pi-zero/>

Pour plus d'information sur le port serie:

<https://www.framboise314.fr/le-port-serie-du-raspberry-pi-3-pas-simple/#Les_UART_du_Raspberry_Pi>

Configuration de la liaison UART

La liaison /dev/ttyAMA0 est configurée pour vous renvoyer un écho de ce qu’elle reçoit. La liaison se configure avec stty.

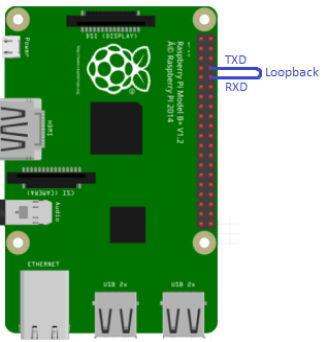
|  |  |
| --- | --- |
| **Etape 1** | Pour connaître votre configuration actuelle utilisez :  ---> stty -F /dev/ttyAMA0 -a |
| **Etape 2** | Pour comprendre précisément la configuration, la **page man de stty** vous sera bien utile. J'ai trouvé la page version français :  <http://manpagesfr.free.fr/man/man1/stty.1.html> |
| **Etape 3** | utilisez la commande suivante pour avoir une configuration fonctionnelle:  ---> sudo stty -F /dev/ttyAMA0 115200 cs8 -cstopb -onlcr -echo -echoe -echok -opost |

Voici deux commandes pour manipuler la liaison serie avec le terminal

|  |  |
| --- | --- |
| Commande 1 | Pour lire ce qui est reçu dans la liaison:  ---> cat /dev/ttyAMA0 |
| Commande 2 | Pour écrir dans la liaison  ---> echo -e -n "Bonjour\x00" > /dev/ttyAMA0  **-e** Demande à la fonction echo d’interpréter les caractère spéciaux, ce qui transformera notre ’\x00’ en caractère de fin de chaîne.  **-n** Demande à la fonction echo de ne pas générer automatiquement de retour à la ligne. |

<http://poivron-robotique.fr/Liaison-UART-du-Raspberry-Pi.html>

Teste du port série pour valider le fonctionnement

On va faire ce qu'on appelle un Loopback. On va relier les ports GPIO 8 et 10 correspondants à TXD et RxD (Données émises = Données reçues).  
Lorsqu'on va envoyer des données sur le port série TxD, elles vont revenir par le port RxD.

|  |  |
| --- | --- |
| **Etape 1** | **Téléchargement de minicom**  Connectez-vous en terminal (en mode texte ou ouvrez un terminal sur le bureau)  ---> sudo apt-get update  Puis  ---> sudo apt-get install minicom |
| **Etape 2** | **Connexion sur la liaison serie avec minicom**  ---> minicom -D /dev/serial0 -b 115200  -D permet de definir le port  -b permet de definir la vitesse de transmission |
| **Etape 3** | **Verification**  Si tu tapes quelque chose tu devrais avoir un echo. |

Après vérification, il faut enlever le câble qui relie TXD à RXD !!!!

## Annexes 2: Ficher SerialObj.h

#ifndef SERIAL\_H\_

#define SERIAL\_H\_

#include <termios.h>

#include <sys/types.h>

#include <string>

using namespace std;

#define SERIAL\_DEFAULT\_SPEED 115200

#define SERIAL\_DEFAULT\_PARITY 'n'

#define SERIAL\_DEFAULT\_BIT\_DATA 8

#define SERIAL\_DEFAULT\_BIT\_STOP 1

#define SERIAL\_DEFAULT\_FLUX 0

#define SERIAL\_NO\_FILE (-1)

#define SERIAL\_STATUS\_ERROR (-1)

#define SERIAL\_STATUS\_INIT 0

#define SERIAL\_STATUS\_READY 1

//#define SERIAL\_STATUS\_OPEN 2

#define READ\_MAX 20

class SerialObj

{

private:

char \* file\_name;

FILE \* file;

int fd;

int speed;

int parity;

int nbr\_bits\_data;

int nbr\_bits\_stop;

int flux;

struct termios config;

struct termios save;

int read\_max;

int status;

public:

/\* Constructor and Destructor\*/

SerialObj(char \* file\_name);

~SerialObj();

/\*Methodes\*/

void com\_serial\_init(char \* file\_name);

void com\_serial\_exit();

void com\_serial\_open();

void com\_serial\_open\_dynamixel(float baudrate);

void com\_serial\_close();

void com\_serial\_config\_save();

void com\_serial\_config\_restaure();

int com\_serial\_read\_buff(char \* buffer, int size);

int com\_serial\_write\_buff(char \* buffer, int size);

int com\_serial\_write\_buff\_float(float nbr);

int print\_Arduino\_buff(char\* tab);

int write\_Arduino\_buff(string msg);

void process(char ID\_Commande,int Data);

int com\_serial\_read\_buff\_wait(char \* buffer, int size, uint usec);

int com\_serial\_write\_buff\_wait(char \* buffer, int size, uint usec);

int com\_serial\_fprintf(char \* fmt, ...);

int com\_serial\_fscanf(char \* fmt, ...);

void com\_serial\_flush(int mode);

//Setters

void com\_serial\_set\_speed(int speed);

void com\_serial\_set\_parity(int parity);

void com\_serial\_set\_nbr\_bits\_data(int nbr\_bits\_data);

void com\_serial\_set\_nbr\_bits\_stop(int nbr\_bits\_stop);

void com\_serial\_set\_flux(int flux);

void com\_serial\_set\_blocking\_mode();

void com\_serial\_set\_non\_blocking\_mode();

void com\_serial\_set\_config();

//Getters

int com\_serial\_get\_speed();

int com\_serial\_get\_parity();

int com\_serial\_get\_nbr\_bits\_data();

int com\_serial\_get\_nbr\_bits\_stop();

int com\_serial\_get\_flux();

int com\_serial\_get\_status();

//Others

void com\_serial\_print\_config();

};

#endif /\* SERIAL\_H\_ \*/

## Annexes 3: Ficher SerialObj.cpp

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <string.h>

#include <string>

#include <fcntl.h>

#include <termios.h>

#include <linux/serial.h>

#include <sys/ioctl.h>

#include <sys/time.h>

#include <sys/types.h>

#include <errno.h>

#include <stdarg.h>

#include "commande.h"

#include "SerialObj.h"

using namespace std;

/\*Constructor and Destructor\*/

SerialObj::SerialObj(char \* file\_name){

com\_serial\_init(file\_name);

com\_serial\_flush(TCIOFLUSH);

com\_serial\_open();

if(this->status == SERIAL\_STATUS\_ERROR){

printf("erreur d'ouverture !\n");

com\_serial\_exit();

}

}

SerialObj::~SerialObj(){

}

/\*Methodes\*/

void SerialObj::com\_serial\_init(char \* file\_name){

if (file\_name == NULL) {

return;

}

this->file\_name = file\_name;

this->fd = SERIAL\_NO\_FILE;

this->speed = SERIAL\_DEFAULT\_SPEED;

this->parity = SERIAL\_DEFAULT\_PARITY;

this->nbr\_bits\_data = SERIAL\_DEFAULT\_BIT\_DATA;

this->nbr\_bits\_stop = SERIAL\_DEFAULT\_BIT\_STOP;

this->flux = SERIAL\_DEFAULT\_FLUX;

this->read\_max = READ\_MAX;

this->status = SERIAL\_STATUS\_INIT;

}

void SerialObj::com\_serial\_exit() {

if (this->fd >= 0) {

com\_serial\_config\_restaure();

com\_serial\_close();

}

this->status = SERIAL\_STATUS\_INIT;

}

void SerialObj::com\_serial\_open() {

if (this->file\_name == NULL) {

return;

}

/\*

if (serial->status != SERIAL\_STATUS\_READY) {

return;

}

\*/

//this->fd = open(this->file\_name, O\_RDWR | O\_NOCTTY | O\_SYNC | O\_NONBLOCK);

this->fd = open(this->file\_name, O\_RDWR | O\_NONBLOCK);

if (this->fd < 0) {

this->status = SERIAL\_STATUS\_ERROR;

return;

}

com\_serial\_set\_config();

//To FILE \* p

this->file = fdopen(this->fd, "r+");

if (this->file == NULL) {

close(this->fd);

this->status = SERIAL\_STATUS\_ERROR;

return;

}

this->status = SERIAL\_STATUS\_READY;

usleep(1000000);

}

void SerialObj::com\_serial\_open\_dynamixel(float baudrate) {

struct termios termios\_dynamixel;

struct serial\_struct serial\_info;

int retval;

int i\_baudrate = (int) baudrate;

if (this->file\_name == NULL) {

return;

}

this->fd = open(this->file\_name, O\_RDWR | O\_NOCTTY | O\_SYNC);

//this.fd = open(serial->file\_name, O\_RDWR | O\_NOCTTY | O\_SYNC | O\_NONBLOCK);

if (this->fd < 0) {

this->status = SERIAL\_STATUS\_ERROR;

return;

}

memset(&termios\_dynamixel, 0, sizeof(struct termios));

termios\_dynamixel.c\_cflag = B38400|CS8|CLOCAL|CREAD;

termios\_dynamixel.c\_iflag = IGNPAR;

termios\_dynamixel.c\_oflag = 0;

termios\_dynamixel.c\_lflag = 0;

termios\_dynamixel.c\_cc[VTIME] = 0;

termios\_dynamixel.c\_cc[VMIN] = 0;

tcflush(this->fd, TCIFLUSH);

tcsetattr(this->fd, TCSANOW, &termios\_dynamixel);

retval = ioctl(this->fd, TIOCGSERIAL, &serial\_info);

if(retval < 0) {

close(this->fd);

return;

}

serial\_info.flags &= ~ASYNC\_SPD\_MASK;

serial\_info.flags |= ASYNC\_SPD\_CUST;

serial\_info.custom\_divisor = serial\_info.baud\_base / i\_baudrate;

retval = ioctl(this->fd, TIOCSSERIAL, &serial\_info);

if(retval < 0) {

close(this->fd);

return;

}

com\_serial\_close();

this->fd = open(this->file\_name, O\_RDWR | O\_NOCTTY | O\_SYNC | O\_NONBLOCK);

//this.->fd = open(serial->file\_name, O\_RDWR | O\_NOCTTY | O\_SYNC);

if (this->fd < 0) {

this->status = SERIAL\_STATUS\_ERROR;

return;

}

termios\_dynamixel.c\_cflag = B38400|CS8|CLOCAL|CREAD;

termios\_dynamixel.c\_iflag = IGNPAR;

termios\_dynamixel.c\_oflag = 0;

termios\_dynamixel.c\_lflag = 0;

termios\_dynamixel.c\_cc[VTIME] = 0;

termios\_dynamixel.c\_cc[VMIN] = 0;

cfsetispeed(&termios\_dynamixel, i\_baudrate);

cfsetospeed(&termios\_dynamixel, i\_baudrate);

tcflush(this->fd, TCIFLUSH);

tcsetattr(this->fd, TCSANOW, &termios\_dynamixel);

this->status = SERIAL\_STATUS\_READY;

}

void SerialObj::com\_serial\_close() {

if (this->fd< 0) {

return;

}

close(this->fd);

this->fd = SERIAL\_NO\_FILE;

}

void SerialObj::com\_serial\_config\_save() {

memcpy(&this->save,&this->config,sizeof(struct termios));

}

void SerialObj::com\_serial\_config\_restaure() {

int retval;

if (this->fd < 0) {

return;

}

retval = tcsetattr(this->fd, TCSANOW, &this->save);

if (retval < 0) {

this->status = SERIAL\_STATUS\_ERROR;

}

}

int SerialObj::com\_serial\_read\_buff(char \* buffer, int size) {

int nbr\_read;

int nbr\_to\_end = size;

char \* buf = buffer;

int cpt\_read = 0;

while (nbr\_to\_end) {

nbr\_read = read(this->fd, buf, nbr\_to\_end);

if (nbr\_read < 0) {

if (errno != EAGAIN) {

this->status = SERIAL\_STATUS\_ERROR;

printf("COM READ ERROR - %d\n", errno);

return nbr\_read;

}

}

if (nbr\_read == 0) { /\* Test #Read max \*/

cpt\_read++;

if (cpt\_read > this->read\_max) {

printf("COM READ MAX\n");

return nbr\_read;

}

}

if (nbr\_read > 0) {

cpt\_read = 0; /\* RAZ \*/

}

nbr\_to\_end -= nbr\_read;

buf += nbr\_read;

}

return size;

}

int SerialObj::com\_serial\_write\_buff(char \* buffer, int size) {

int nbr\_write = 0;

nbr\_write = write(this->fd, buffer, size);

return nbr\_write;

}

int SerialObj::com\_serial\_write\_buff\_float(float nbr){

char buffer[4]={0};

int nbr\_write;

memcpy(buffer,&nbr,4);

nbr\_write=com\_serial\_write\_buff(buffer,4);

return nbr\_write;

}

void SerialObj::process(char id\_commande, int data){

char buff[4]={START\_BYTE,id\_commande};

memcpy(buff+2,&data,2);

com\_serial\_write\_buff(buff,4);

}

int SerialObj::print\_Arduino\_buff(char\* tab){

char buffer[2];

int length=0;

int cpt=0;

int nb\_read=0;

int i=0;

while(nb\_read!=1){

nb\_read=com\_serial\_read\_buff\_wait(buffer+i,1,5000);

if(\*(buffer+i)>='0' && \*(buffer+i)<='9'){

i++;

nb\_read=0;

}

}

length= atoi(buffer);

while(cpt<length){

nb\_read = com\_serial\_read\_buff\_wait(buffer,1,5000);

if(nb\_read==1 && \*buffer>31){

\*(tab)=\*buffer;

tab++;

cpt++;

}

}

//while(nb\_read==1){

//if(com\_serial\_read\_buff\_wait(buffer,1,5000)==1);

// com\_serial\_flush(TCIOFLUSH);

//}

return length;

}

int SerialObj::write\_Arduino\_buff(string msg){

com\_serial\_flush(TCOFLUSH);

const char\* ptr = NULL;

int taille=0;

int nb\_write=0;

taille=msg.length();

ptr = msg.c\_str();

nb\_write=com\_serial\_write\_buff((char\*)ptr,taille);

return nb\_write;

}

int SerialObj::com\_serial\_read\_buff\_wait(char \* buffer, int size, uint usec\_wait) {

struct timeval time\_wait;

fd\_set Set;

int rc;

int nbr\_read = 0;

FD\_ZERO(&Set);

FD\_SET(this->fd, &Set);

time\_wait.tv\_sec = 0;

time\_wait.tv\_usec = usec\_wait;

rc = select(FD\_SETSIZE, &Set, NULL, NULL, &time\_wait);

if (rc != 1) {

return 0;

}

if (FD\_ISSET(this->fd, &Set)) {

nbr\_read = com\_serial\_read\_buff(buffer, size);

}

return nbr\_read;

}

int SerialObj::com\_serial\_write\_buff\_wait(char \* buffer, int size, uint usec\_wait) {

struct timeval time\_wait;

fd\_set Set;

int rc;

int nbr\_write = 0;

FD\_ZERO(&Set);

FD\_SET(this->fd, &Set);

time\_wait.tv\_sec = 0;

time\_wait.tv\_usec = usec\_wait;

rc = select(FD\_SETSIZE, NULL, &Set, NULL, &time\_wait);

if (rc != 1) {

return 0;

}

if (FD\_ISSET(this->fd, &Set)) {

nbr\_write = com\_serial\_write\_buff(buffer, size);

}

return nbr\_write;

}

int SerialObj::com\_serial\_fprintf(char \* fmt, ...) {

va\_list ap;

int n;

va\_start(ap, fmt);

n = vfprintf(this->file, fmt, ap);

va\_end(ap);

return n;

}

int SerialObj::com\_serial\_fscanf(char \* fmt, ...) {

va\_list ap;

int n;

va\_start(ap, fmt);

n = vfscanf(this->file, fmt, ap);

va\_end(ap);

return n;

}

void SerialObj::com\_serial\_flush(int mode) {

if (mode & TCIFLUSH) {

tcflush(this->fd, TCIFLUSH);

}

if (mode & TCOFLUSH) {

tcflush(this->fd, TCOFLUSH);

}

if (mode & TCIOFLUSH) {

tcflush(this->fd, TCIOFLUSH);

}

}

/\*Setters\*/

void SerialObj::com\_serial\_set\_speed(int speed){

this->speed = speed;

}

void SerialObj::com\_serial\_set\_parity(int parity){

this->parity = parity;

}

void SerialObj::com\_serial\_set\_nbr\_bits\_data(int nbr\_bits\_data){

this->nbr\_bits\_data = nbr\_bits\_data;

}

void SerialObj::com\_serial\_set\_nbr\_bits\_stop(int nbr\_bits\_stop){

this->nbr\_bits\_stop = nbr\_bits\_stop;

}

void SerialObj::com\_serial\_set\_flux(int flux){

this->flux = flux;

}

void SerialObj::com\_serial\_set\_blocking\_mode(){

fcntl(this->fd,F\_SETFL,fcntl(this->fd,F\_GETFL)&~O\_NONBLOCK);

}

void SerialObj::com\_serial\_set\_non\_blocking\_mode(){

fcntl(this->fd,F\_SETFL,fcntl(this->fd,F\_GETFL)|O\_NONBLOCK);

}

void SerialObj::com\_serial\_set\_config() {

int retval;

tcgetattr(this->fd, &this->config);

com\_serial\_config\_save();

cfmakeraw(&this->config);

switch(this->speed) {

case 50:

cfsetspeed(&this->config,B50);

break;

case 75:

cfsetspeed(&this->config,B75);

break;

case 110:

cfsetspeed(&this->config,B110);

break;

case 134:

cfsetspeed(&this->config,B134);

break;

case 150:

cfsetspeed(&this->config,B150);

break;

case 200:

cfsetspeed(&this->config,B200);

break;

case 300:

cfsetspeed(&this->config,B300);

break;

case 600:

cfsetspeed(&this->config,B600);

break;

case 1200:

cfsetspeed(&this->config,B1200);

break;

case 1800:

cfsetspeed(&this->config,B1800);

break;

case 2400:

cfsetspeed(&this->config,B2400);

break;

case 4800:

cfsetspeed(&this->config,B4800);

break;

case 9600:

cfsetspeed(&this->config,B9600);

break;

case 19200:

cfsetspeed(&this->config,B19200);

break;

case 38400:

cfsetspeed(&this->config,B38400);

break;

case 57600:

cfsetspeed(&this->config,B57600);

break;

case 115200:

cfsetspeed(&this->config,B115200);

break;

default:

this->status = SERIAL\_STATUS\_ERROR;

return;

}

switch(this->parity) {

case 'n':

this->config.c\_cflag &= ~ PARENB;

break;

case 'p':

this->config.c\_cflag |= PARENB;

this->config.c\_cflag &= ~ PARODD;

break;

case 'i':

this->config.c\_cflag |= PARENB;

this->config.c\_cflag |= PARODD;

break;

default:

this->status = SERIAL\_STATUS\_ERROR;

return;

}

this->config.c\_cflag &= ~CSIZE;

switch(this->nbr\_bits\_data) {

case 5:

this->config.c\_cflag |= CS5;

break;

case 6:

this->config.c\_cflag |= CS6;

break;

case 7:

this->config.c\_cflag |= CS7;

break;

case 8:

this->config.c\_cflag |= CS8;

break;

default:

this->status = SERIAL\_STATUS\_ERROR;

return;

}

if (this->nbr\_bits\_stop == 1)

this->config.c\_cflag &= ~ CSTOPB;

else

this->config.c\_cflag |= CSTOPB;

//this.config.c\_cflag &= ~ CLOCAL;

//this.config.c\_cflag |= (CLOCAL | CREAD);

//this.config.c\_cflag |= HUPCL;

//this.config.c\_cflag |= CRTSCTS;

this->config.c\_cflag |= CREAD;

this->config.c\_iflag = IGNPAR | IGNBRK;

switch(this->flux) {

case 1:

this->config.c\_iflag |= IXON | IXOFF;

break;

case 2:

this->config.c\_cflag |= CRTSCTS;

break;

default:

this->config.c\_cflag |= CLOCAL;

break;

}

this->config.c\_oflag &= ~OPOST;

this->config.c\_lflag = 0;

this->config.c\_cc[VMIN] = 1;

this->config.c\_cc[VTIME] = 10;

//printf("Config After :\n");

//print\_termios(&this.config);

retval = tcsetattr(this->fd, TCSANOW, &this->config);

if (retval < 0) {

this->status = SERIAL\_STATUS\_ERROR;

return;

}

tcflush(this->fd, TCOFLUSH);

tcflush(this->fd, TCIFLUSH);

return;

}

/\*Getters\*/

int SerialObj::com\_serial\_get\_speed(){

return this->speed;

}

int SerialObj::com\_serial\_get\_parity(){

return this->parity;

}

int SerialObj::com\_serial\_get\_nbr\_bits\_data(){

return this->nbr\_bits\_data;

}

int SerialObj::com\_serial\_get\_nbr\_bits\_stop(){

return this->nbr\_bits\_stop;

}

int SerialObj::com\_serial\_get\_flux(){

return this->flux;

}

int SerialObj::com\_serial\_get\_status() {

return this->status;

}

//Others

void SerialObj::com\_serial\_print\_config() {

//int i;

printf("Config.c\_iflag = %u\n", this->config.c\_iflag);

printf("Config.c\_oflag = %u\n", this->config.c\_oflag);

printf("Config.c\_cflag = %u\n", this->config.c\_cflag);

printf("Config.c\_lflag = %u\n", this->config.c\_lflag);

//printf("Config.c\_line = [%c]\n", this.config.c\_line);

//printf("Config.c\_cc = ");

/\*

for (i=0; i < NCCS; i++) {

printf(" [%2x]", this.config.c\_cc[i]);

}

printf("\n");

\*/

}

## Annexes 4: Fonction process()

int process(VideoCapture& capture)

{

float w = capture.get(CAP\_PROP\_FRAME\_WIDTH);

float h = capture.get(CAP\_PROP\_FRAME\_HEIGHT);

cout << "Width : " << w << " Height : " << h << endl;

Point pt1\_s( 0\*w, 0.1\*h), pt1\_e( 1\*w, 0.1\*h); // line 1

Point pt2\_s( 0\*w, 0.2\*h), pt2\_e( 1\*w, 0.2\*h); // line 2

/ Point pt3; // line 3

Point center1(0.0f, 0.0f); // line center 1

Point center2(0.0f, 0.0f); // line center 2

Point center3(0.0f, 0.0f); // line center 3

int n = 0;

char filename[200];

string window\_name = "video | q or esc to quit";

cout << "press space to save a picture. q or esc to quit" << endl;

namedWindow(window\_name, WINDOW\_KEEPRATIO); //resizable window;

Mat frame, frame\_gray, dst;

for (;;)

{

capture >> frame;

if (frame.empty())

{

cout << " ... empty frame " << endl;

break;

}

// Visu only : BGR -> BW

cvtColor( frame, frame\_gray, COLOR\_BGR2GRAY );

// Visu only : TH

threshold( frame\_gray, dst, th, 255, 0 );

// Compute

Vec3b buf;

LineIterator it1(frame, pt1\_s, pt1\_e, 8);

int sum1 = 0;

center1 = Point(0,0);

for(int i = 0; i < it1.count; i++, ++it1)

{

buf = frame.at<Vec3b>(it1.pos());

if( (buf.val[0] + buf.val[1] + buf.val[2])/3 < th )

{

center1 += it1.pos();

sum1 ++;

}

}

if(sum1 > 0) center1 /= sum1;

LineIterator it2(frame, pt2\_s, pt2\_e, 8);

int sum2 = 0;

center2 = Point(0,0);

for(int i = 0; i < it2.count; i++, ++it2)

{

buf = frame.at<Vec3b>(it2.pos());

if( (buf.val[0] + buf.val[1] + buf.val[2])/3 < th )

{

center2 += it2.pos();

sum2 ++;

}

}

if(sum2 > 0) center2 /= sum2;

cout << " center 1 ( " << center1.x << ", " << center1.y << ") : "<< sum1 << endl;

cout << " center 2 ( " << center2.x << ", " << center2.y << ") : "<< sum2 << endl;

// Visu only : line with arrow

// arrowedLine(frame\_gray, pt1\_s, pt1\_e, Scalar(0, 255, 0), 2 );

arrowedLine(dst, center1, center2, Scalar(255, 255, 255) , 2 );

imshow(window\_name, dst);

char key = (char)waitKey(30); //delay N millis, usually long enough to display and capture input

switch (key) {

case 'q':

case 'Q':

case 27: //escape key

cout << " ... Stopping " << endl;

return 0;

case ' ': //Save an image

sprintf(filename,"filename%.3d.jpg",n++);

imwrite(filename,dst);

cout << "Saved " << filename << endl;

break;

default:

break;

}

}

return 0;

}

}