Revue de projet

Notre objectif est de réaliser un programme d'entraînement pour les sprinters de haut niveau. Le système que nous allons vous montrer va servir de “lièvre” en reproduisant, si possible à l’identique, les performances de la meilleure course d’Usain Bolt sur 100m et 200m.

1. Protocoles de mesures, cahier des charges (écart 1)
2. Protocole écarts 1 et 2 (J et R) vitesse

**J**:Pour vérifier la concordance entre la réalité et le cahier des charges, nous avons mis en place différents protocoles pour mesurer la vitesse maximum de la voiture.

**R**: Pour cela, nous avons réalisé l’expérience suivante (photo).

**J**: Nous avons dans un premier temps utiliser la formule v=d/t, ceci nous donnait uniquement la vitesse moyenne sur la distance parcourue et non la vitesse maximum. Alors nous avons modifié le protocole et fait démarrer la voiture 20m avant le plot pour être sûr d’obtenir une moyenne de la vitesse max.

**R**: Ensuite, nous avons décidé d’utiliser la vidéo de la seconde expérience pour obtenir la courbe de la vitesse du véhicule en fonction du temps. Nous avons donc traité la vidéo avec le logiciel Aviméca. Après avoir étalonné les distances, nous avons pointé les positions de la voiture en fonction du temps, puis nous avons envoyé le pointage sur regressi (voir photo). Enfin, il nous a suffit de regarder la vitesse maximum atteinte par la voiture. Cependant, Il faut prendre en compte que la vidéo utilisée est légèrement courbée sur les côtés ce qui fausse très légèrement les résultats obtenus.

**J:** Pour finir, nous voulions vérifier l’exactitude de nos valeurs, alors nous avons profité de l’expérience faite par E et Adam pour l’accélération pour vérifier une dernière fois les valeurs obtenues, nous avons utilisé leur résultat pour calculer la vitesse max. Enfin, pour calculer l’écart 1, il nous a suffit de faire la moyenne des données obtenue par les différents protocoles.

1. Protocole accélération avec acquisition capteur Einstein (Adam et E)

**E:** Afin de vérifier la valeur de l’accélération du véhicule fournie par le cahier des charges, nous avons dû mettre en place un protocole expérimental et ainsi utiliser le capteur Einstein.

**Adam: (Slide Schéma)** En effet grâce au Einstein Labmate (petite boîtier blanc) et à son module accéléromètre nous pouvons effectuer l'acquisition de l’accélération en temps réel sur la tablette Einstein en se connectant directement sur le Labmate en Bluetooth.

Comme on peut le voir sur ce schéma, la tablette Einstein est connectée en bluetooth au module. Ainsi, après l’acquisition nous importons sur l’ordinateur le fichier .csv et nous le traitons sur le logiciel Excel.

**E: (Slide Photo 1)** Avant de réaliser ceci nous avons effectué des tests sur le capteur et l’avons étalonné à l’aide du logiciel du constructeur sur ordinateur. **(Slide Photo 2)** Par la suite nous l’avons disposé sur la voiture comme vous pouvez voire sur cette (photo).Enfin après avoir effectué un ensemble d’acquisition d’accélération de la voiture sur une distance d’une vingtaine de mètres, nous avons obtenu un ensemble de fichier .csv que nous avons transféré sur l’ordinateur.

**Adam:(Slide Graph )** Une fois sur l’ordinateur nous nous sommes rendus compte qu’un nombre trop important de mesures avait été fait, rendant les fichiers .csv beaucoup trop lourds à manipuler et pour les filtrer par exemple. Nous avons donc pris une mesure sur 10.

**E :** Un autre problème c’est alors posé : le nombre de mesures étant trop importantes un bruit conséquent était présent dans le signal : nous avons donc dû le filtrer numériquement. Nous avons ainsi appliqué un filtrage sur Excel à l’accélération afin d’obtenir ce résultat (graphique).

**Adam:** Une fois l’accélération filtrée obtenue, on peut déterminer l’accélération maximum de la voiture et en intégrant celle-ci on obtient enfin la vitesse en fonction du temps. Ceci nous a alors permis d’obtenir la vitesse maximum du véhicule.

1. Protocole de la dérive et masse de la voiture

**R:** Les derniers points du cahier des charges à vérifier sont la dérive du véhicule et la masse.

**J:** La dérive c’est la distance séparant le trajet initial de la voiture et le trajet réalisé par la voiture, en d’autres termes, c’est la déviation du véhicule.

**R:** Pour cela, nous avons utilisé les lignes des carreaux du sol de la salle, nous avons pris une distance de 10m en ligne droite et fait avancer la voiture sur cette droite à sa vitesse minimum après avoir réglé au mieux la dérive.

**J:** Quand elle a atteint les 10m, nous avons mesuré la distance séparant la ligne droite et la position du véhicule (voir schéma).

**R:** Pour mesurer la masse du véhicule, il suffit de le peser sans la batterie.

1. Conclusion et écart

**J :** Voici les résultats pour les différents protocoles de mesure de la vitesse max:

-Vmax1=29.232 km/h (v=d/t, 3 mesures) -Vmax2=34.5±0.3km/h (protocole vidéo)

-Vmax3=29.628 km/h (protocole acc) -Vmoy=31.14km/h

Ecart=30.8% (voir tableau)

**R :** Cet écart est trop important, nous n'atteindrons jamais la vitesse maximum d’Usain Bolt avec cette voiture. Il nous est donc impossible de reproduire la course de l’homme le plus rapide du monde (voir simu).

Concernant la dérive, notre expérience a démontrée une dérive bien plus importante que celle annoncée dans le cahier des charges :

*Valeur constructeur : < 4cm / 10 mètres.*

*Valeur obtenue : 19,5 cm / 10 mètres.*

Ce qui nous donne un écart de 387,5 %, toutefois rien d’affolant, nous nous attendions à un résultat très élevé. De plus, cette dérive va pouvoir être corrigée sur le projet final à l’aide de la caméra.

**J:** Enfin, la masse du modèle réduit sans la batterie. Avec une masse constructeur annoncée à 1200 g, notre 1109 g relevé se rapproche plus ou moins de cette valeur, avec un écart de 7,6%. En ce qui concerne la masse, le cahier des charges est respecté.

1. Modélisation et amélioration
2. Modélisation, protocole et écart 3 (J et R)

**R :** Après avoir mené les différentes expériences pour vérifier les valeurs du cahier des charges, il nous restait à modéliser notre produit pour obtenir l’écart 2 et 3. Nous avons fait cela à l’aide du logiciel MatLab Simulink. Le professeur nous a donné un modèle que nous avons dû compléter **(diapo)**.

**J: (PAS CHANGER DIAPO)**Nous avons débuté en relevant le rayon de la roue du Buggy télécommandé. Ensuite, à l’aide des différentes docs techniques fournies, nous avons cherché à obtenir le rapport de transmission.

*Calcul du rapport :*

*==0.10096.*

**R :** Par la suite, nous avons réalisé des protocoles afin de trouver I0 (voir slide schéma) et le calcul de Ωmoteur (avec le rapport de transmission et comptant le nbr de tour effectués par le moteur pour un tour de roue). A l’aide de ces données nous avons pu obtenir le reste des valeurs qu’il nous manquait en appliquant les formules apprises en cours: la constante de f.e.m, le couple Cpertes, le couple d'arrachement (Breakaway friction torque) et le coefficient de frottements visqueux (viscous friction coefficient). (Affichage tableau Excel avec tous les résultats).

A l’aide de ces données, nous avons complété les différents bloc comme vous pouvez voir sur le diapo (2 slides)

**J:** Notre modèle a été validé par le professeur, cependant ce n’était que le modèle à vide, ce qui veut dire que d’après celui-ci la voiture ne subit aucune force. Nous avons donc dû rechercher et calculer les différentes forces qu’il fallait ajouter à notre modèle.

Les forces subies par la voiture sont donc les forces de frottement avec l’air que l’on va nommer Faero et la force résistance au roulement Frr.

**R :** Dans le logiciel Matlab, il y a déjà un bloc prévu pour Faero, il faut juste le compléter avec le coefficient de traînée et la surface de projection du véhicule S, en contact même avec les frottements de l’air.

Pour calculer la surface S, il a fallu dans un premier temps déterminer les différentes parties du véhicule qui seraient “en contact” avec le flux d’air. On obtient donc la figure ci-contre(diapo avec la figure). puis nous avons mesuré les différentes distances et on a obtenu la surface S=0.02198 m²

**J:** Ensuite, après de multiple recherche sur internet nous avons décidé de choisir un coefficient de traînée d’après les différents Cx des différents type de voiture trouvé sur internet, notamment en comparant avec le Cx d’une voiture de course qui est en moyenne aux alentours de 0.25 et celui d’une citadine qui est de 35 à 45. Nous obtenons donc Cx=0.3.

Enfin, nous avons utilisé un bloc de Matlab que le professeur avait prévu pour cet effet, nous l’avons complété avec les différentes données.

Pour finir avec la modélisation, nous avons fait une expérience avec la voiture pour obtenir la force de résistance au roulement. Pour cela nous avons dû démonter les roues et les remonter à l’envers pour ne pas prendre en compte le réducteur. Enfin, nous avons tiré la voiture avec un dynamomètre de 5N et nous avons obtenu Frr=0.7

Nous avons complété le bloc avec Frr et connecté les deux blocs à la chaîne comme on peut voir sur la capture d’écran.

**CCl R écart 2.** **Cl J écart 3.**

1. Amélioration : accélération et suivi de lignes (Adam et E)

**Adam:(Slide Graph Usain Bolt)** Afin d’améliorer la voiture nous avons tout d’abord dû essayer de reproduire les courbes d’accélération d’un 100m effectué par Usain Bolt (graph) en adaptant avec les performances de notre voiture. Pour cela il fallait dans un premier temps pouvoir contrôler le variateur de la voiture à l’aide d’une carte Arduino Uno dans laquelle on enverrait un programme spécifique.

**E: (Slide Code + photos)** Une fois la carte branchée au variateur, on y exporta un programme qui nous permettait d’envoyer une valeur au variateur du véhicule comprise entre 89 et 124, valeur faisant varier la vitesse de rotation des roues arrière. Nous avons donc placé la voiture sur le banc de test et fait des essais.

**Adam:(Slide Tachymètre)** À l'aide d’un Tachymètre nous avons relevé la vitesse de rotation des roues arrière à vide correspondant à la valeur envoyée par l’Arduino. **(graph N tr/min en fonction de la commande).**

**E:** A partir de ce moment là, nous avions 2 possibilités : l’une étant de définir la commande a envoyer en fonction de la vitesse dans un intervalle, ce qui est facile à mettre en place mais peu précis ou alors de trouver la formule mathématiques permettant de passer d’une vitesse en km/h en commande au servo-moteur. **(Slide flèche)** C’est la deuxième option que nous avons choisie car elle nous paraissait plus adaptée au besoin de l’utilisateur : une vitesse précise.

**Adam:(Slide Graph Excel)** Après avoir essayé de nombreuses fonctions de références, il nous fallut utiliser l’outil sur Excel permettant de trouver la fonction de tendance entre deux grandeurs : ici les tr/min en fonction de la commande au servomoteur. Nous avons ainsi choisi d’utiliser une fonction polynomiale du 3e degré (graph).

**E:(Slide Arduino)** Une fois cette fonction trouvée nous avions juste à l’inclure directement dans notre programme arduino : nous avons ainsi créé une fonction de type flottant avec en paramètre la vitesse en km/h. Elle nous permet ainsi de convertir à tout moment dans notre programme et nous permettra plus tard de convertir la vitesse rentré par l’utilisateur en commande.

**Adam:**La fonction convert se décompose en trois étapes: On calcule d’abord la vitesse de rotation en rad/s puis en tr/min; Ensuite on utilise la fonction du 3e degré trouvé sur Excel pour retrouver la commande du servomoteur; Enfin on renvoie cette valeur retrouvée. **(Graph 3 phases)** Grâce à notre phase d’études des caractéristiques réelles de la voiture nous avons créé un parcours détaillé et réalisable par celle-ci en fonction du schéma et de l’allure de celui d’Usain Bolt. Une problématique s’est alors posée : Quelles valeurs devront rentrer les coureurs? Pour simplifier cette partie nous avons rentré directement la vitesse atteinte à la fin de chaque phase et la distance effectuée.

**E:(Code)** Dans notre boucle principale nous avons ainsi procédé sur chaque phase. Prenons ici la phase 1 par exemple : nous rentrons le temps de la phase et la vitesse atteinte à la fin de cette phase en km/h. On appelle la fonction convert avec la vitesse rentré que l’on stocke directement dans une variable : C1. Nous calculons ensuite l’incrémentation de la commande moteur pour chaque tour de boucle. Nous avons régler chaque tour de boucle avec un délai de 10 ms: nous divisons donc la différence commande entre la base (89 correspondant à l’arrêt) et la commande finale nous la divisons par le temps nous obtenons l’incrémentation par seconde : on divise par 100 pour obtenir l’incrémentation pour 10ms. Ainsi on crée une boucle for allant de 89 à la valeur finale et on vient l’envoyer au servomoteur appelé “myservo” et on l’affiche dans la console et on attend un délai de 10 ms.

**Adam:** En utilisant 3 fois cette fonction et en configurant les variables pour chacune des phases on obtient ainsi les 3 phases d’accélérations d’un coureur. **(Photo montage)** Pour vérifier que ces 3 phases étaient bien respecté selon le parcour préétablie nous avons utilisé le banc d’essai et récupéré sur une autre carte arduino la fréquence de rotation à l’aide d’un branchement avec un pont diviseur et un programme Arduino.**(Conv flèche)**On peut ensuite en déduire la valeur de la rotation omega (w=F\*2π) en rad/s puis la vitesse en m/s (V=R\*w). **(Graph vérification)** Cependant on trouve un rapport de deux entre les valeurs retrouvées et les valeurs réelles : nous avons donc décidé de multiplier les valeurs par 2 arbitrairement. On obtient ainsi cette courbe.

**E:** **(Graph superposition)**On peut voir que l’allure générale de la courbe respecte bien le parcours préconfiguré et en superposant on voit que même si l’allure générale est respectée il reste surtout dans la phase de démarrage une allure qui diffère de l’original. Le problème provient du variateur : à basse vitesse nous avons remarqué que la vitesse de rotation est instable et a tendance à vite augmenter alors qu’on remarque qu’une fois une certaine vitesse atteinte on retrouve bien des droites et donc des représentations de fonction affine dans les phase 2 et 3.

**Adam: (Photos pixy)**Nous sommes ensuite passé sur le suivi des lignes à l’aide de la reconnaissance par vidéo et de la caméra Pixy2. La caméra Pixy2 est un module compatible Arduino et Raspberry Pie qui permet de faire du traitement vidéo en temps réel grâce à une IA pré-entraînée et présentant 2 fonctionnalités : color tracking (suivi de couleurs) et line tracking (suivi de vecteurs). La fonctionnalitée nous intéressant étant le suivi de couleur la ligne au sol étant trop épaisse pour être considéré comme un vecteur.

**E:** Nous l’avons d’abord placé sur le système “pan-tilt” fourni avec la caméra : cela nous permettra de tester la caméra et de pouvoir une fois sur la voiture régler l’angle de la caméra.

**Adam:(Photos pixymon + zumo)** Pour la suite du projet nous nous sommes fortement basés sur un projet basique de la caméra Pixy2 : zumo. Pour commencer à manipuler la caméra il suffit ainsi de régler une couleur sur Pixymon le logiciel du constructeur et des rectangles apparaissent sur les objets détectés, ceci délimite en fait les zones établies par les couleurs préprogrammées sur les objets possédant ses dites couleurs. **(Gif demo)** A l’aide du programme pan-tilt on obtient ainsi un suivi d’un objet par la caméra (gif) et on remarque que celle-ci est très réactive : ce qui devrait faciliter la phase de suivi de lignes.

**E:** **(Code 1)** Nous sommes ensuite rentrés dans la phase de programmation de l’Arduino, en nous basant encore une fois sur le code exemple du zumo. On retrouve ainsi 2 phases : tout d’abord la recherche d’un objet préconfiguré sur Pixymon si la variable index = -1, une fois l’objet trouvé on appelle la fonction trackblock qui vient nous donner les coordonnées sur l’image de l’objet reconnu. **(Code 2)** Ensuite, tant que l’objet représenté par la variable block est toujours détecté, on vient effectuer des rotations à la caméra grâce à la variable global panOffset qui change selon la position du bloc (sur l’axe x). On vient ensuite maper cette valeur pour qu’elle corresponde à une valeur que les servomoteurs des roues avant : de -100 à 100 à entre 50 et 140 selon nos tests. On vient ensuite envoyer cette valeur dans les servomoteurs qui feront tourner le véhicule. Si aucun bloc n’a été détecté, on recommence la phase de recherche.

1. A venir (Application + essais)

**Adam:(Gif résultat)** En combinant les deux précédents programmes nous avons ainsi pu effectuer des essais sur piste. Comme on peut le voir sur ces images (gif) à basse vitesse le suivi est assez bien effectué même si on peut voir qu’à la fin un tracé sur le côté fera changé sa trajectoire.

**E:** Il y a donc plusieurs pistes d’améliorations :

* créer un test détectant les mauvaises détections : si l’angle de rotation est trop important c’est que l’objet détecté n’est pas la ligne à suivre
* venir incrémenter l’angle jusqu’à être parfaitement aligné car actuellement le changement d’angle est trop brutal
* Prendre en compte la vitesse de la voiture en effet à haute vitesse les à-coups sont trop importants
* Réduire les fréquence de recherche car les servomoteurs ne devrait pas changer la trajectoire en permanence et ainsi empêcher les déplacement et rotations inutile car il ne faut pas oublier que sur un 100 mètres la piste est droite et le but est d’uniquement corriger la dérive de la voiture.

**Adam:** Il faut donc rendre la voiture plus fiable pour l’utilisateur et améliorer ses trajectoires.**(Screen app)**  Nous avons commencé à travailler parallèlement à ses améliorations dont la fonctionnalité est d’améliorer grandement l’expérience utilisateur : l’application mobile. En effet le but à terme est d’avoir une application fonctionnelle se connectant en bluetooth à la voiture et permettant à l’utilisateur de paramétrer ses phases de courses. Un apk en alpha (juste graphique) est déjà disponible et en voici quelques captures d’écrans.