RAPPORT DE PROJET
LASER XY

Mathieu STEPHAN, Rémi BIN, Lise TALBOTIER

22 juin 2007

Table des matières

Remerciements

Nous tenons à remercier tout spécialement,
(Laboratoire ELMI)
M Piau
M Buez
M Vinnatier
M Perreira
M Imbert
M Latorre
M Blanchard
(Laboratoire A2SI)
M Grandpierre
M Couprie
(General Scanning)
M Stockhausen
M Roehrl
(Logic PD)
M LaBorde
(Analog Devices)
M Erskine

Introduction Ce projet est une réalisation étudiante dans le cadre du cursus ESIEE de 3° année en lien avec les laboratoires A2SI et ELMI. Il a pour but la modélisation de formes 3D par un laser optique. Ce projet s'axe autour de trois domaines distincts : l'asservissement, l'électronique numérique et analogique et l'informatique. Chaque personne travaillant sur ce projet s'orientera en septembre 2007 vers des majeures différentes. L'objectif sous-jacent du projet est alors de coordonner un travail d'équipe et de réussir à faire cohabiter des savoir-faire différents.

Première partie

ASSERVISSEMENT

CONTROLE DES GALVANOMETRES

1 GALVANOMETRE

Pour dessiner des figures sur une surface, nous devons orienter très rapidement des moteurs spécifiques sur lesquels sont montés des miroirs pour orienter le faisceau laser. Ces moteurs sont des galvanomètres M2 9,5mm. Ils utilisent la technologie des aimants mobiles. C'est le même phénomène lorsque nous plaçons une boussole à l'intérieur d'un enroulement de spires. Lorsque nous faisons passer du courant dans les spires, cela fait tourner le rotor du galvanomètre. Ces types de galvanomètres sont très rapides et ont un faible moment d'inertie ce qui leur confèrent les bonnes caractéristiques pour effectuer un balayage angulaire très rapide pour dessiner tous les points de chaque figure.

Le type de galvanomètre que nous possédons fonctionne en open-loop. C'est à dire que sans boucle de retour, la réponse de position du galvanomètre à une entrée indicielle serait une droite qui irait jusqu'à saturation. En effet, le galvanomètre recevrait une impulsion et irait jusqu'en butée, d'où la nécessité de faire un asservissement de position des galvanomètres.

2 MODELISATION

Pour pouvoir asservir nos galvanomètres, nous devons les modéliser sachant que la fonction de transfert est tenue confidentielle par les fabricants des galvanomètres que nous utilisons. D'après la datasheet des galvanomètres, nous avons décidé de les modéliser sous forme de moteur à courant continu. Tout d'abord parce qu'ils s'alimentent par du courant. De plus, nous possédons la résistance et l'inductance d'enroulement ainsi que le moment d'inertie et la constante de couple.

Calcul de la fonction de transfert en boucle ouverte du moteur à courant continu grâce aux équations électriques et mécaniques :
Soit :
U : tension d'entrée du moteur
E : force électromotrice
C_m : couple moteur
k_c : constante de couple
k_e : constante électrique
J : moment cinétique
On a :
$U = E + RI + L\frac{di}{dt}$
Nous négligeons l'inductance L, car elle n'intervient que dans la constante de temps élec-

trique qui est négligeable par rapport à la constante de temps mécanique.
$E = k_e\Omega$
$C_m = k_c I$
Nous supposons $k_c = k_e = k$.
$J\frac{d\Omega}{dt} = C_m$
$U = k\Omega + R\frac{J}{k}p\Omega$

$$\Rightarrow H(p) = \begin{cases} \frac{\Omega}{U} \\ \frac{1}{k+\frac{RJ}{k}p} \\ \frac{\frac{1}{k}}{1+\frac{RJ}{k^2}p} \end{cases} \text{ avec } T_M = \frac{RJ}{k^2} \text{ : constante de temps mécanique}$$

Cette fonction de transfert nous donne la vitesse angulaire en fonction de la tension d'entrée. Nous souhaitons faire un asservissement de position donc nous recherchons la fonction de transfert en boucle ouverte du galvanomètre. Le galvanomètre est constitué d'un capteur de position dont la sortie est envoyée sur un circuit qui est réglé pour varier de + ou - 1,5V pour $+15^\circ$ à -15°. De plus, notre consigne étant fournie entre +10V et -10V, nous devons rajouter un gain qui par la suite permettra de comparer la position à la consigne sur la même plage de variation. Ce qui nous donne comme modélisation totale du galvanomètre le schéma ci-dessous.

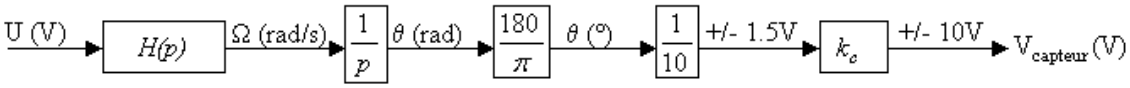
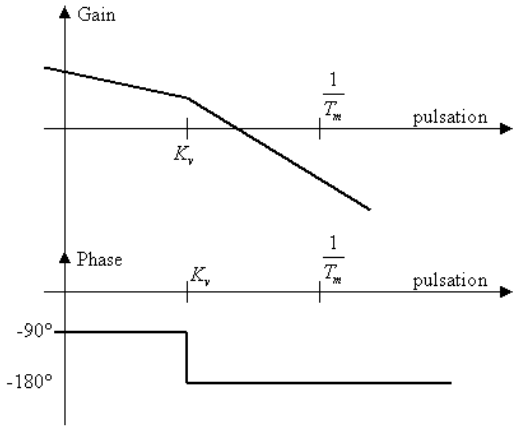


Schéma bloc en boucle ouverte du galvanomètre

Notre fonction de transfert du galvanomètre en boucle ouverte s'écrit donc :

$$F(p) = \begin{cases} H(p) \cdot \frac{1}{p} \cdot \frac{180}{\pi} \cdot \frac{1}{10} \cdot K_c \\ \frac{K_v}{(1+T_M p)p} \end{cases} \text{ avec } K_v = \frac{18K_c}{\pi k}.$$

Nos obtenons donc une fonction de transfert du deuxième ordre avec les diagrammes de Bode ci-dessous.



Diagrammes de Bode de $F(p)$

Ces diagrammes de Bode nous montrent que la phase n'atteindra jamais -180° donc le système est instable ce qui va être corrigé par l'action intégrale du correcteur. De plus, la transmittance du système bouclé peut se mettre sous la forme : $A_R = \frac{K_v}{K_v+p+T_M p^2}$. Avec cette équation, nous obtenons que le coefficient d'amortissement $m = \frac{1}{2\sqrt{T_M K_v}} \simeq 0.3$. m est donc inférieur à 1 donc le système est oscillant ce que nous cherchons à éviter pour notre application.

3 ASSERVISSEMENT DE POSITION

3.1 CHOIX DU CORRECTEUR

Pour obtenir de belles animations, non déformées, il faut que notre asservissement de position soit rapide, précis et stable. Nous souhaitons aussi que l'erreur de position soit nulle ainsi que l'erreur de traînage. Ces critères nous ont conduit à choisir un correcteur Proportionnel Intégrale Dérivée, PID.

L'action proportionnelle permet d'augmenter la rapidité et la précision du système mais diminue sa stabilité. Lorsque nous appliquons une action intégrale cela élimine l'erreur statique et une deuxième action intégrale élimine l'erreur de traînage. L'action intégrale augmente la précision mais diminue la stabilité. L'action dérivée va permettre d'augmenter la stabilité et la rapidité et ainsi amortir rapidement les oscillations dues à une variation subite de l'entrée. Par contre, l'inconvénient de l'action dérivée, c'est qu'elle va amplifier les bruits hautes fréquences, il faut donc rajouter un filtre passe-bas pour filtrer ce bruit.

Nous utilisons les actions du correcteur PID en parallèle. Le schéma de notre correcteur avec filtre sera donc de la forme :

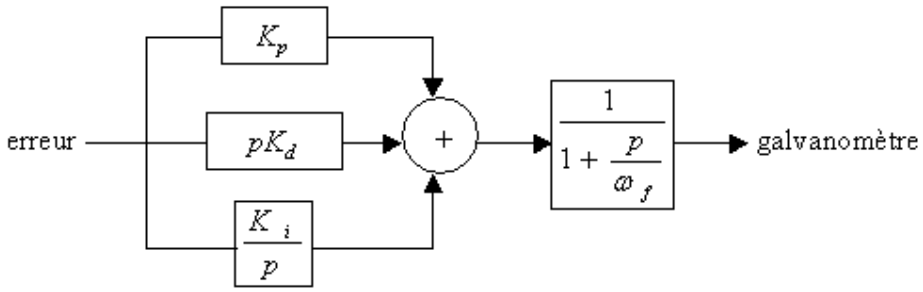


Schéma bloc du correcteur

Avec comme fonction de transfert :

$$C(p) = \begin{cases} (K_p + \frac{K_i}{p} + K_d p) \cdot \frac{1}{1+\frac{p}{\omega_f}} \\ \frac{K_p p + K_i + K_d p^2}{p(1+\frac{p}{\omega_f})} \end{cases}$$

Nous obtenons donc comme système total le schéma bloc ci-dessous.

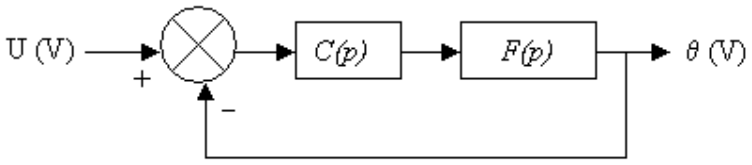


Schéma bloc du système avec correcteur

3.2 CALCUL DES CONSTANTES

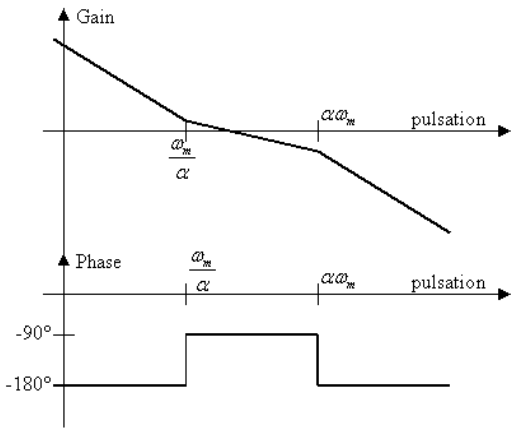
Notre modèle de référence en boucle fermée sera un ordre 3 qui possèdera une erreur de traînage nulle, ce qui implique que la fonction de transfert doit posséder le même coefficient de degré 1 au numérateur et au dénominateur. La fonction la plus simple est celle avec un numérateur d'ordre 1 et un dénominateur de degré 3 car nous recherchons les coefficients du correcteur PID avec filtre. Elle est donc de la forme :

$H^3(p) = \frac{1+n_1 p}{1+n_1 p+d_2 p^2+d_3 p^3}$

Pour déterminer le PID, nous devons réaliser l'équivalence :

$H(p) = \frac{C(p)F(p)}{1+C(p)F(p)} \equiv H_3(p) = \frac{T_3(p)}{1+T_3(p)}$ soit $C(p)F(p) \equiv T_3(p) = \frac{H_3(p)}{1-H_3(p)} = \frac{1+n_1 p}{p^2(d_2+d_3 p)}$ Nous proposons l'écriture suivante : $T_3(p) = \frac{G\omega_m^{2(1+\frac{\alpha p}{\omega_m})}}{p^2(1+\frac{p}{\alpha\omega_m})}$

Le diagramme de Bode de cette fonction donne :



Diagrammes de Bode de $T_3(p)$

La phase est maximale pour la pulsation ω_m. Pour assurer la stabilité maximale, nous rendons la marge de phase maximale avec $|T_3| = 1$ et nous avons $G = \frac{1}{\alpha}$. Le système en boucle fermée possède alors la fonction de transfert :

$$H_3(p) = \frac{1+\alpha\frac{p}{\omega_m}}{1+\alpha\frac{p}{\omega_m}+\frac{p^2}{G\omega_m^2}+\frac{p^3}{\alpha G\omega_m^3}}$$

Ce système est caractérisé par des abaques qui donnent le dépassement indiciel et le temps de réponse à 5% en fonction de α. Grâce à ces abaques, nous choisissons $\alpha = 20$ ce qui correspond à un dépassement minimal de 4%. Pour un α égale à 20, nous obtenons un temps de réponse à 5% de 2.4, $tr\omega_m \approx 2.4$. Nous déterminons ensuite le correcteur $C(p)$ selon la relation $C(p)F(p) \equiv T_3(p)$. Ce qui dans notre cas donne :

$$C(p) = \frac{K_p p+K_i+K_d p^2}{p(1+\frac{p}{\omega_f)}} = \frac{G\omega_m^2(1+\frac{\alpha p}{\omega m})(1+T_M)}{K_v(1+\frac{p}{\alpha\omega_m})p}$$

Par identification, nous obtenons :
$\omega_f = \alpha\omega_m$, $K_i = \frac{G\omega_m^2}{K_v}$, $K_p = K_i(T_M + \frac{\alpha}{\omega_m})$, $K_d = K_i\frac{\alpha T_M}{\omega_m}$.
Nous avons calculé les constantes pour différentes valeurs du temps de réponse. Sur la datasheet des galvanomètres, nous avons un temps de réponse optimal de 1.5ms pour un angle de 0.4° ce qui correspond au maximum à la différence entre 2 points d'une figure. Nous avons donc calculé les constantes pour le temps de réponse optimal et le temps de réponse optimal divisé par 2 si nous souhaitons accélérer le système. Comme nous avons fait une approximation du modèle et s'il ne répond pas exactement comme nous l'avons prévu, par exemple un dépassement indiciel trop important au départ, nous avons choisi un temps de réponse maximal qui correspond au temps de réponse de la boucle ouverte du galvanomètre T_M divisé par 2. Nous obtenons comme constante avec ces différents temps de réponse :

Si $tr = 1.5ms \Rightarrow \omega_m \simeq 1607rad/s \Rightarrow \omega_f \simeq 32157rad/s \Rightarrow G = 0.5 \Rightarrow K_i \simeq 78Hz \Rightarrow K_p \simeq 1.1 \Rightarrow K_d \simeq 1.65ms$

Si $tr = 0.75ms \Rightarrow \omega_m \simeq 3218rad/s \Rightarrow \omega_f \simeq 64315rad/s \Rightarrow G = 0.5 \Rightarrow K_i \simeq 311Hz$

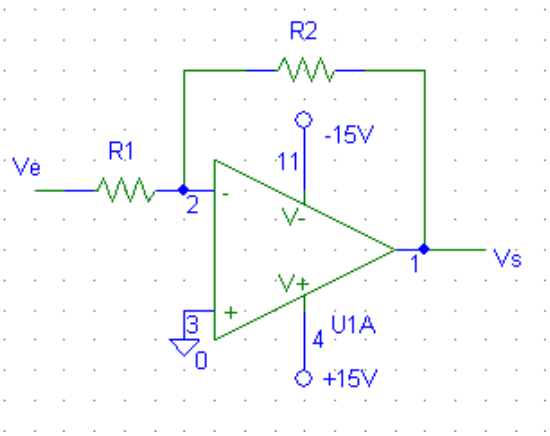
$\Rightarrow K_p \simeq 2.5 \Rightarrow K_d \simeq 3.3ms$

Si $tr = \frac{T_M}{2} = 2.5515ms \Rightarrow \omega_m \simeq 945rad/s \Rightarrow \omega_f \simeq 18905rad/s \Rightarrow G = 0.5 \Rightarrow K_i \simeq 26.9Hz \Rightarrow K_p \simeq 0.615 \Rightarrow K_d \simeq 0.968ms$

3.3 CALCUL DES COMPOSANTS PHYSIQUES DU CORRECTEUR

Ces constantes de temps ainsi calculées, vont nous permettre de dimensionner nos composants. En effet, chaque action du correcteur est faite à l'aide d'un circuit composé d'un amplificateur opérationnel.

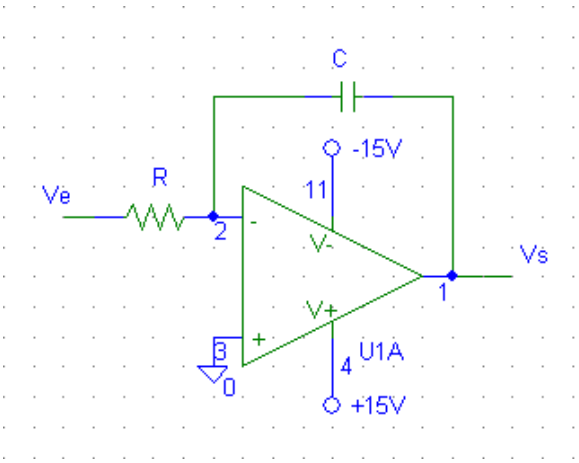
Pour l'action proportionnelle :



Montage amplificateur inverseur

$\frac{V_s}{V_e} = -\frac{R_2}{R_1}$.
K_p varie de 0.615 à 2.5 $\Rightarrow R_1 = 22k\Omega$ et R_2 varie de 13.5kΩ à 54.3kΩ.

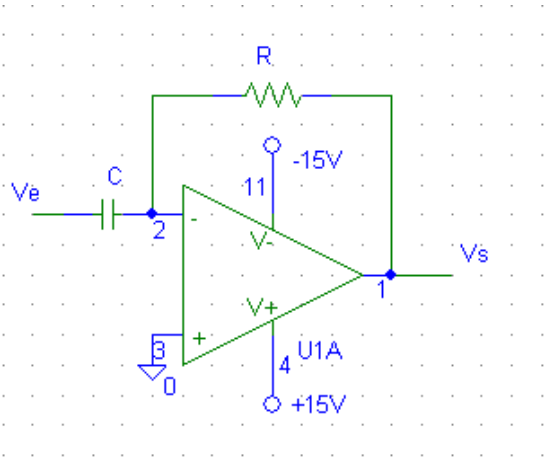
Pour l'action intégrale :



Montage intégrateur

$\frac{V_s}{V_e} = -\frac{1}{RCp}$.
RC varie de 3.2 ms à 37.2 ms $\Rightarrow C = 1\mu F$ et R varie de 3.2kΩ à 37.2kΩ.

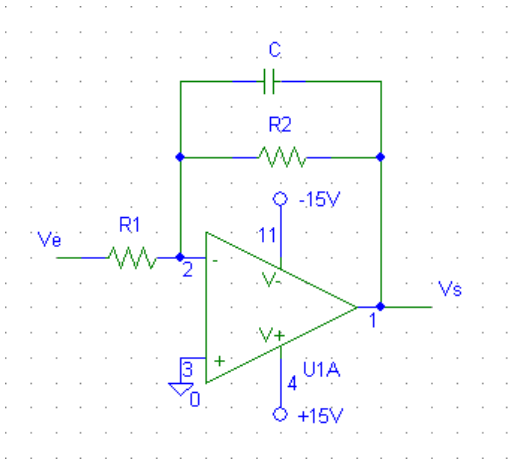
Pour l'action dérivée :



Montage dérivateur

$\frac{V_s}{V_e} = -RCp$.
K_d varie de 0.968 ms à 1.65 ms $\Rightarrow C = 0.1\mu F$ et R varie de 9.6kΩ à 16.5kΩ.

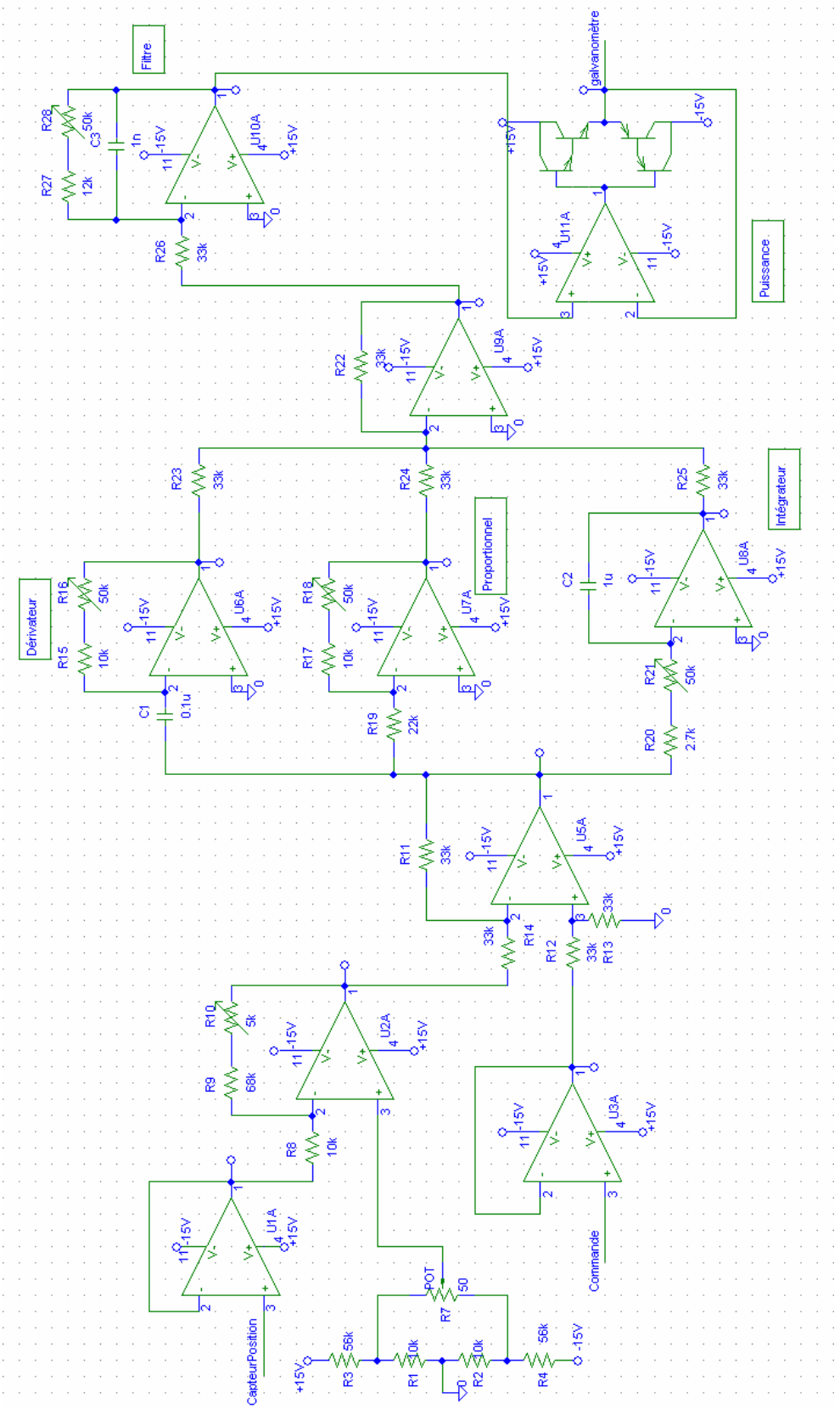
Pour le filtre passe-bas :



Filtre passe-bas

$\frac{V_s}{V_e} = -\frac{\frac{R_2}{R_1}}{1+R_2Cp}$.
R_2C varie de $1.56 \cdot 10^{-5}$ à $5.3 \cdot 10^{-5} \Rightarrow C = 1nF$ et R_2 varie de 15.55kΩ à 52.9kΩ.

Pour les variations des résistances dans les différents montages, nous utilisons des potentiomètres multitours. Au final, le schéma de notre asservissement pour un galvanomètre donne :


Filtre
R28
50k
C3
1n
R27
12k
R26
33k
U10A
-15V
+15V
galvanomètre
U11A
Puissance
R22
33k
U9A
R23
33k
R24
33k
R25
33k
Intégrateur
Dérivateur
R16
50k
R15
10k
C1
0.1u
U6A
Proportionnel
R18
50k
R17
10k
U7A
C2
1u
U8A
R21
50k
R19
22k
R20
2.7k
R11
33k
U5A
R14
33k
R12
33k
R13
33k
R10
5k
R9
68k
R8
10k
U2A
U3A
U1A
Commande
CapteurPosition
POT
R7
50
R3
56k
R1
10k
R2
10k
R4
56k

Nous avons mis en parallèle les actions proportionnel, intégrale et dérivée appliquées à l'erreur, la différence entre la commande et le capteur de position du galvanomètre. Pour le capteur de position, après un circuit adapté qui convertit les ampères délivrés par le capteur en une tension comprise entre +1,5V et -1,5V nous avons mis un amplificateur opérationnel avec un gain et un offset qui nous permettra de régler la plage de variation du capteur si elle n'est pas exactement entre + ou -1,5V. Pour l'offset, nous avons donc réalisé un schéma qui nous délivre du + ou -1,5V. Pour la partie puissance, nous avons utilisé un montage push-pull avec un amplificateur opérationnel qui annule la distorsion de courant. De plus nous avons monté les transistors en Darlington pour avoir plus d'intensité car l'amplificateur opérationnel en sortie en délivre très peu. Ce montage de puissance permet d'alimenter le galvanomètre en courant sous 1A continu et 5A pic. Pour respecter ces contraintes, nous utilisons des transistors NPN BD243 et PNP BD243. Pour tout ce montage, nous utilisons des amplificateurs OP213 qui sont des amplificateurs de précision avec peu de bruit. Ils possèdent aussi un produit gain-bande important de 3.5MHz et un slew rate de $0.9V/\mu s$.

4 PROTOCOLE D'AJUSTEMENT DU CORRECTEUR

4.1 PROTOCOLE D'AJUSTEMENT DU CORRECTEUR

Nous avons réalisé la carte d'asservissement sur circuit imprimé. Nous devons régler la carte via les multitours pour qu'elle corresponde aux conditions réelles de notre système. Pour cela, nous avons suivi le protocole ci-dessous qui est valable pour un galvanomètre. Il faut donc le répéter pour chaque galvanomètre.

Avant de souder les multitours, nous les avons réglés pour qu'ils soient à la valeur optimale du temps de réponse donné par le constructeur, qui est de 1.5ms.

Nous avons relié le capteur de position du galvanomètre à la carte alimentée mais avec une commande d'entrée reliée à la masse et la sortie de la carte non branchée au galvanomètre. Ensuite, nous avons fait bouger le galvanomètre à la main et vérifier que la réponse du capteur de position était correcte : en sortie du suiveur après le capteur de position, puis après le gain et enfin à l'entrée du galvanomètre. Ces tests nous permettent d'assurer que le capteur de position fonctionne et que notre carte d'asservissement répond bien.

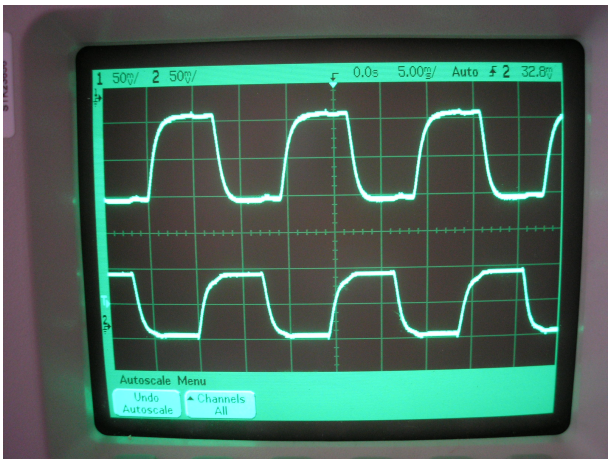
Pour régler l'offset du capteur de position, il faut mettre la commande à la masse, ajuster le spot du laser pour qu'il soit au centre de la figure optique grâce au potentiomètre de l'offset. La réponse des deux galvanomètres est visible en branchant les capteurs de position sur l'oscilloscope en mode XY. Ensuite, nous ajustons la plage de variation du capteur de

position entre + ou - 10V.

Pour régler le correcteur, il faut connecter la carte au galvanomètre puis appliquer une commande en créneaux à une fréquence de 50Hz et d'amplitude 0.2V. Ensuite, nous visualisons la réponse du galvanomètre et il faut régler les potentiomètres du dérivateur et du proportionnel, de telle sorte que le signal se rapproche le plus d'un créneau. Pour notre utilisation, nous devons minimiser au maximum les dépassements mais aussi essayer d'avoir un temps de réponse qui se rapproche le plus de 1ms. Pour régler l'erreur, il faut visualiser l'erreur de position et faire varier le potentiomètre de l'intégrateur pour que la partie plate de l'erreur soit nulle.

4.2 TESTS DE L'ASSERVISSEMENT

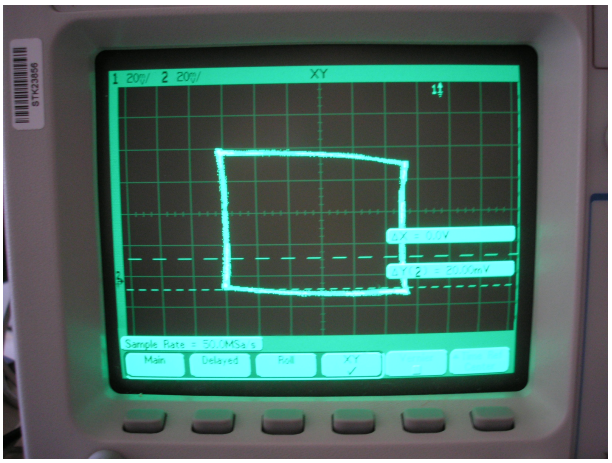
Pour tester la carte, nous avons suivi le protocole décrit ci-dessus en faisant bien attention à ce que nos galvanomètres ne chauffent pas. Lors du réglage du créneau, nous avons obtenu le relevé ci-dessous.



Relevé des détecteurs de position des galvanomètres sous 70Hz

Nous avons donc bien minimisé le dépassement. Nous avons du faire un compromis entre dépassement et temps de réponse. En effet, avec les réglages optimaux, nous avions un dépassement sur la position basse. Pour le corriger, nous avons du augmenter notre temps de montée égale à 3ms, sachant que notre temps de descente est de 2ms.

Lorsque nous positionnons l'oscilloscope en mode XY nous obtenons la figure ci-dessous, qui représente exactement ce que forme le faisceau laser.



Carré obtenu en mode XY

Cette figure est légèrement déformée, cela est du aux temps de réponse des galvanomètres qui ne sont pas optimaux.

Deuxième partie

CNA ET CONTROLE

Troisième partie

INFORMATIQUE ET CONTROLE

1 GESTION DU LASER

1.1 COUNTER ET TEMPS D'EXECUTION

1.1.1 Utilité

Le processeur ARM7 est fréquencé à 74Mhz. Cette fréquence est beaucoup trop elevée pour l'envoi de donnée au Convertisseur Numérique Analogique. L'objectif est d'envoyer les coordonnées des points successifs toutes les milisecondes. Il est donc nécessaire d'employer un compteur pour retarder l'envoi de l'information d'une part, et pour envoyer cette information de manière régulière.

1.1.2 Configuration

L'ARM7 dispose de trois compteurs. Le premier (timer0) contient cinq registres de capture et deux registres de comparaison alors que les deux autres timer ne disposent que de deux registres de capture et de deux registres de comparaison. Comme le timer1 est utilisé, nous réserverons le timer2 pour l'envoi des informations vers le bus de données à destination du Convertisseur Numérique Analogique. Pour configurer le timer, il suffit de suivre la procédure de la documentation en paramètrant les registres selon nos souhaits. La procédure est la suivante :

```
Timer2_CTRL &= ~(0x3);                  // Stop et Clear Timer 2
Timer2_CMP1 = 0x4570;                   // Valeur de comparaison = 1ms
Timer2_CTRL |= ((1 << 13) | (011<<2));  // Compteur à ZERO quand
                                        // COUNTER = COMPARE1
                                        // Clock division =12
(...)
Timer2_CTRL |= 2;                       // Démarrage du timer 2
(...)
Timer2_STATUS = 4;                      // Reset des interruptions
```

Dans la configuration choisie, il faut 12 cycles d'horloge processeur contre un seul pour le timer. Quand le timer atteint la valeur 0x4570, une interruption est automatiquement générée par le timer.

Avant de traiter les interruptions, il est bon de savoir que les interruptions, pour le timer2, peuvent être générées soit par une capture, soit par une comparaison, soit par un overflow. Il n'existe en fait qu'une seule interruption pour chaque timer qui résulte d'un "OU" entre ces différentes interruptions internes. Comme nous n'utiliserons que le registre de comparaison 1, il faut inhiber les interuptions provenant des autres éléments.

```
Timer2_INT_CTRL = 4;                    // Interruption sur COMP1 uniquement
```

1.2 INTERRUPTIONS

1.2.1 Différence IRQ/FIQ et Choix

L'ARM7 est pourvu d'un contrôleur d'interruptions appellé "Vectored Interrupt Controller" (VIC). Le VIC possède 32 sources d'interruptions dont 23 interruptions internes et 2 interruptions logicielles. Parmi ces interruptions, on peut utiliser pour notre timer des interruptions vectorisées.
Les interruptions internes sont répertoriées dans une table. Dans notre cas, l'interruption correspondant au timer2 est l'interruption numéro 6. Il est alors nécessaire de rediriger cette interruption vers une fonction de notre choix (adresse de l'ISR : Interrupt Service Routine).

1.2.2 Configuration et priorités

Il est aussi nécessaire de lui associer un niveau de priorité. Ce niveau de priorité se définit par un chiffre entre 0 et 15 ; plus le chiffre est petit, plus l'interruption est prioritaire. Dans notre cas, nous choisissons arbitrairement un niveau de 4 (les niveaux précédents étant occupés par des interruptions plus prioritaires). D'après la documentation, nous obtenons ces quelques lignes :

```
VIC_IntSelect &= ~(1 << TIMER2_INT);    // Timer 2 interrupt is an IRQ interrupt
VIC_VectAddr4 = (unsigned int)timer2ISR; // adresse ISR pointe vers la fonction
                                        // timer2ISR()
VIC_VectCtrl4 = 0x20 | TIMER2_INT;      // Interruption de niveau 4
VIC_IntEnable |= 1 << TIMER2_INT;       // Autorisation des interruptions pour
                                        // le timer2
(...)
VIC_VectAddr = 0;                       // Met à jour la table des priorités
                                        // lors de l'interruption
```

1.2.3 Tests

Nous avons récupérer des librairies liée au microprocesseur développée par Mathieu Stéphan lors d'un précédent stage. Dans cette librairie, il y avait une fontion permettant de faire clignoter des LED du kit de développement. En utilisant cette fonction, nous avons pu vérifier le fonctionnement de notre timer et de notre interruption. Nous avons aussi pu régler précisément à l'osciloscope la valeur du timer pour obtenir 1 ms de période.

1.3 CONFIGURATION SMC

Nous avons vérifier le bon fonctionnement de notre timer et de notre interruption. Il s'agit maintenant d'avoir un bus de donnée pourvu des signaux CHIP SELECT, READ, WRITE pour pouvoir communiquer avec le CNA. Il faut pour cela configurer le Static Memory Controller. Ce controler permet d'interfacer des composants externes au microcontrôleur. Le SMC propose quatre bancs de communication. Comme le 1° banc est utilisé par la ROM, nous utiliserons le banc 2 pour communiquer avec le CNA. L'adresse de base du SMC est 0x40000000, et pour le banc 1, c'est 0x44000000. Il faut ensuite configurer le SMC pour notre CNA. Le bus d'entrée du CNA étant de 12 bits, il faut envoyer l'information sur 16 bits et non sur 8 bits. Les 4 premiers bits seront alors dépourvus d'information. Nous configurons aussi tous les temps d'attente au maximum, car le temps d'envoi de l'information sera de toute façon limitée par le compteur.

2 MOTEUR 3D

2.1 STRUCTURES DE DONNEES

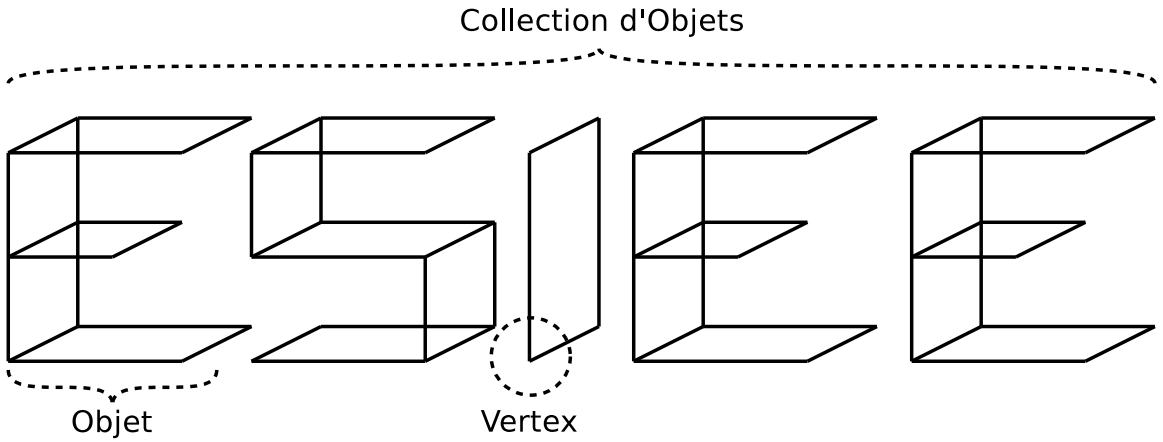
2.1.1 Vertex, Objets et Collection d'Objet

Le projet s'axe autour de trois structures de données que sont les Vertex, les Objets et les Collections d'Objets.

Le Vertex, ou sommets en français, correspondent aux coordonnées de chaque point de la forme que l'on souhaite affichée. Ces point présentent 4 coordonnées : x, y, z et " laser " qui permet de déterminer les coupures du faisceau afin de pouvoir se déplacer d'un sommet à l'autre sans dessiner de droite entre eux.

Les Objets sont, comme leur nom l'indique, la représentation logicielle des objets 3D. Plus communément, un objet est un ensemble de Vertex (sommets).

Enfin, les Collections d'Objets sont des ensembles d'Objets.



Cette conception structurale peut s'apparentée à celle utiliser dans des librairies 3D comme OPENGL par exemple. Elle permet notamment de pouvoir contrôler indépendamment ou simultanément chaque Vertex, Objets ou Collection d'Objets.

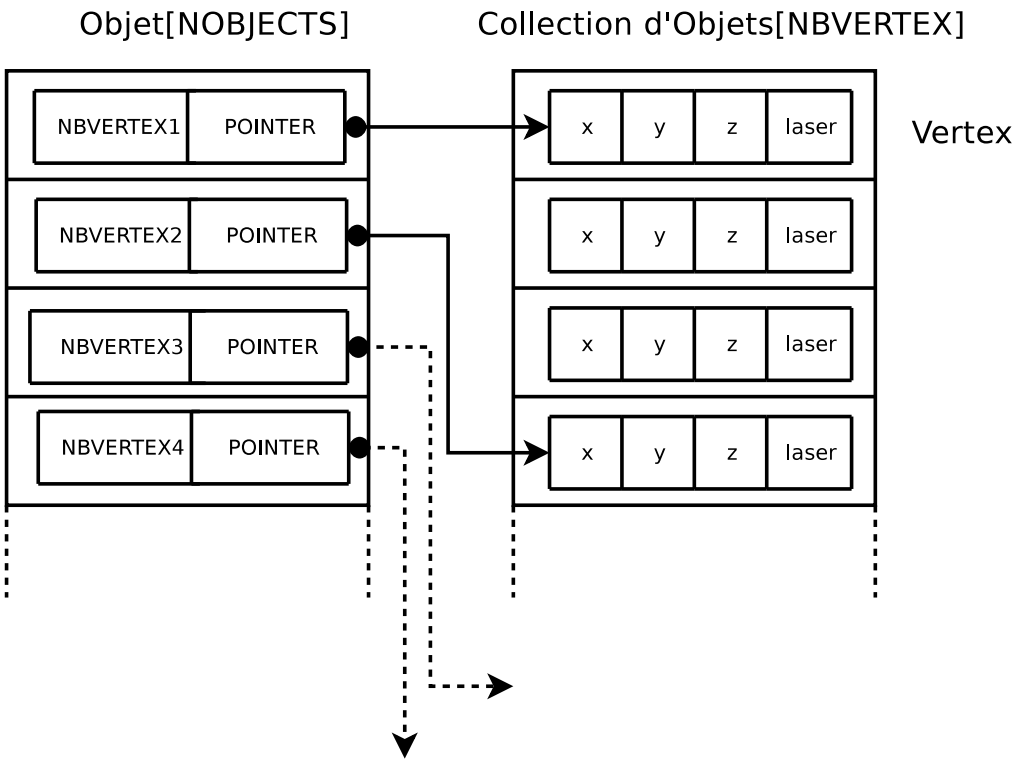
2.1.2 Gestion Mémoire

Le système sur lequel repose notre projet peut s'apparenter à une architecture pour informatique embarquée. Nous sommes à la fois limité par un temps de calcul long et par un faible espace mémoire. L'objectif principal de l'informatique embarquée est alors de réaliser le meilleur compromis entre temps calcul et stockage mémoire.

Comme nous les verrons dans le chapitre III, les données concernant les Vertex et les Objets (donc, les Collections d'Objets) sont stockée en dur dans la mémoire. Aucune allocation dynamique de mémoire n'est réalisée dans le moteur 3D pour deux raisons majeures. La première est qu'il n'existe aucune MMU (Memory Managment Unit) au sein du microcontrôleur, ce qui implique qu'il n'existe pas de protections en écriture sur certaines zones mémoire comme la zone programme par exemple. Les données du TAS peuvent alors être écrites sur la PILE, la zone des variables globales, etc. D'autre part, nous nous sommes aperçu que les fonctions d'allocation mémoire comme " malloc " ou celle codées par le constructeur n'avaient pas non plus de sécurité en écriture. N'ayant pas besoin d'allocation dynamique, nous nous sommes dirigé vers une structure de données minimale. Comme les données sont écrites en mémoire dès le lancement du programme, on peut indiquer au compilateur le nombre d'Objets et le nombre de Vertex que comporte la figure à afficher.

La première version de notre structure de donnée était conçue de la manière suivante : on allouait 50 cases mémoire pour chaque Objet et on faisait pointer la Collection d'Objets vers ces Objets. On abandonna cette conception car la place gaspillée en mémoire était énorme.

Dans la version finale, la Collection d'Objets comporte une liste (de taille prédéterminée) de tous les Vertex, chaque Objets ayant deux propriété : l'adresse du pointeur vers le début de la liste de Vertex lui correspondant dans la Collection d'Objets et le nombre de Vertex que comporte l'Objet. Le gaspillage mémoire est alors nul.



2.2 OPERATIONS MATHEMATIQUES

2.2.1 Barycentre

Les opérations mathématiques constituent un noyau du problème de la 3D. Elles doivent être modulable de telle sorte d'être applicable aux Objets ou aux Collections d'Objets. Afin de simplifier le traitement, toutes les opérations mathématiques se centreront autour du barycentre de l'Objet ou de la Collection d'Objet.

2.2.2 Transformations

Dans les modèles 3D comme OPENGL ou DIRECTX, les opérations mathématiques sont uniquement matricielles. Il est intéressant de remarquer que ces opérations ne s'opèrent pas sur des matrices carrées à trois dimensions mais à quatre. Pourquoi ? Les opérations appliquées sur les objets doivent être linéaire pour assurer leur interopérabilité ; or, seule la translation n'est pas linéaire. On s'aperçoit que si l'on veut obtenir un système linéaire, il suffit d'ajouter une dimension à la translation, donc à chaque matrice et au sommet considéré. On obtient alors ce type d'opérations :

$$\text{Transformation} \cdot \text{Vertex} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ t_{41} & t_{42} & t_{43} & t_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Dans notre cas, il est préférable de ne pas utiliser de matrice afin de ne pas surcharger le microprocesseur. Nous nous contenterons de traduire les matrices.

$$\text{Matrice de translation :} \begin{bmatrix} 1 & 0 & 0 & k_x \\ 0 & 1 & 0 & k_y \\ 0 & 0 & 1 & k_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x + k_x \\ y + k_y \\ z + k_z \\ w \end{bmatrix}$$

$$\text{Matrice d'homothétie} \begin{bmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x \cdot k_x \\ y \cdot k_y \\ z \cdot k_z \\ w \end{bmatrix}$$

$$\text{Matrice de rotation selon l'axe X} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrice de rotation selon l'axe Y $\begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Remarque : Notre gestion barycentrique des Objets nous oblige à procéder à un changement de repère lors des transformations, les opérations que nous obtenons finalement sont :

Translation : $\begin{cases} x' = x + k_x \\ y' = y + k_y \\ z' = z + k_z \end{cases}$

Homothétie : $\begin{cases} x' = (x - barycentreX) \cdot k_x + barycentreX \\ y' = (y - barycentreY) \cdot k_y + barycentreY \\ z' = (z - barycentreZ) \cdot k_z + barycentreZ \end{cases}$

Rotation selon X : $\begin{cases} y' = & ((y - barycentreY) \cdot cosFAST[\theta]) \\ & -((z - barycentreZ) \cdot sinFAST[\theta]) + barycentreY \\ z' = & ((y - barycentreY) \cdot sinFAST[\theta]) \\ & +((z - barycentreZ) \cdot cosFAST[\theta]) + barycentreZ \end{cases}$

Rotation selon Y : $\begin{cases} x' = & ((x - barycentreX) \cdot cosFAST[\theta]) \\ & +((z - barycentreZ) \cdot sinFAST[\theta]) + barycentreX \\ z' = & ((z - barycentreZ) \cdot cosFAST[\theta]) \\ & -((x - barycentreX) \cdot sinFAST[\theta]) + barycentreZ \end{cases}$

2.3 GESTION MEMOIRE, TEMPS DE CALCUL ET OPTIMISATIONS

La structure de donnée a été mise en place, mais nous observons des ralentissements contraignant que nous souhaitons atténuer. Nous nous intéressons alors à réaliser les meilleurs compromis mémoire/calcul en améliorant différentes fonctions ou concepts que nous avons développé ou que nous avons récupéré par le biais de librairies.

2.3.1 Nombre Réels

Comme le microprocesseur est dépourvu d'accélérateur de calcul pour les FLOAT, la première optimisation possible serait d'utiliser des INT au lieu d'utiliser des FLOAT. En effet, un FLOAT est composé d'un bit de signe, d'un exposant et d'une mantisse alors que le INT (sur 32 bits) pourrait être décomposé en un bit de signe, seize bits correspondant à la valeur de l'entier et quinze bits correspondant aux chiffres après la virgule. Il s'agirait alors de manipuler des chiffres à virgule fixe et non flottante comme le FLOAT. Cependant, la manipulation des INT s'avère complexe, les multiplications difficiles et le gain de temps négligeable.

2.3.2 Fonction cissoïdales

Nous avons effectué des tests avec des Objets contenant plus de 1000 sommets. Le constat que nous avons fait est que le temps d'exécution des fonctions sinus et cosinus de la librairie <math.h> était beaucoup trop long, conformément à nos suppositions.

Dans l'optique d'amélioration des temps d'exécution, nous avons envisagé deux autres solutions. La première, est l'utilisation de l'algorithme de CORDIC (COrdiante Rotation DIgital Computer). Il s'agit d'un algorithme de calcul de sinus/cosinus parfaitement adapté aux microcontrôleurs ou aux systèmes dépourvus de fonctions sinusoïdales.

Les tests que nous avons effectués ont révélés que pour 1000 Vertex, tous les algorithmes étaient trop lents. Nous nous sommes axé vers la seconde solution que nous avons adopté, à savoir, l'utilisation de tableau de valeur pour les sinus et cosinus. L'accès à un cosinus (en degré entier) se fait alors en O(1), donc il est optimal. La précision de l'écran LCD nous permet d'affirmer qu'il ne nous est pas nécessaire de descendre au dessous du degré entier. Cependant, on pourrait affiner les calculs en utilisant une fonction linéaire. Par exemple, si on souhaite un angle de 2.5°, on pourrait utiliser une fonction $f(x) = [E(x) - E(x+1)]/2$ où $E(x)$ est la partie entière de x.

2.3.3 Zones mémoires

En étudiant le microcontrôleur, nous nous sommes aperçu que la limitation en vitesse de calcul résidait principalement par le temps d'accès mémoire. En effet, il faut environ 4 cycle du microprocesseur par accès mémoire. En parcourant la documentation, nous avons découvert la possibilité de déplacer des données ou des fonctions dans des parties de la mémoire plus " proche " du microcontrôleur, notamment la zone " .fast " qui correspondrait à une mémoire cache. Nous avons donc déplacé toutes les fonctions de calcul (rotation, translation, homothétie) dans cette zone mémoire.

2.3.4 LCD et fonctions associées

Droite de Bresenham Nous avons récupéré au début de ce projet certaines librairies graphiques permettant le dessin sur l'écran LCD. De ces librairies, nous n'utiliserons que deux fonctions : drawLine() et drawPixel(), la première pour dessiner nos Objets 3D, la seconde pour effectuer des Tests. Nous avons cependant tenu à comprendre la fonction drawLine() dans l'optique de l'optimiser si besoin était. Il s'est avérer que cette fonction utilisait le principe de la droite de Bresenham, solution que nous avions aussi envisagées. Comme une droite passe toujours entre deux pixel (sauf pour $y = x$), il faut choisir le "bon" pixel à allumer pour avoir une droite. Pour réaliser cela, Brésenham se base sur la dérivée d'une droite $\Delta(x) = \frac{y_2 - y_1}{x_2 - x_1}$ et sur la distance qui sépare cette droite du centre des pixels inférieur et supérieur. Une fois cette fonction manipulée pour minimiser le nombre

de calculs, Bresenham fait ainsi un choix entre
- allumer le pixel supérieur
- allumer le pixel inférieur
- se déplacer latéralement d'une case.

Comme la fonction était déjà optimisée, nous n'y avons pas apporté de modifications.

Double-buffering mémoire Le LCD permet de travailler en double-buffering. Ce principe permet d'écrire dans une zone mémoire pendant qu'une seconde est affichée ; ensuite, la première zone mémoire est affichée et la deuxième sert à l'écriture. Entre deux étapes, seul le pointeur du LCD change d'une zone mémoire vers une autre. L'utilisation d'une telle procédure accélère le traitement des informations et évite nomment de voir le rafraîchissement à l'écran.

3 GENERATION DE FORMES 3D

3.1 MODELES SOUS 3DSMAX

3DSMAX est un logiciel de 3D très répandu qui permet notamment la lecture d'un très grand nombre de fichiers 3D. Notre choix pour 3DSMAX se justifie par sa compatibilité avec les autres logiciels et par la possibilité d'exporter les scènes sous forme de fichiers ASCII : les fichiers .ASE

3.2 ALGORITHME ET GRAPHES

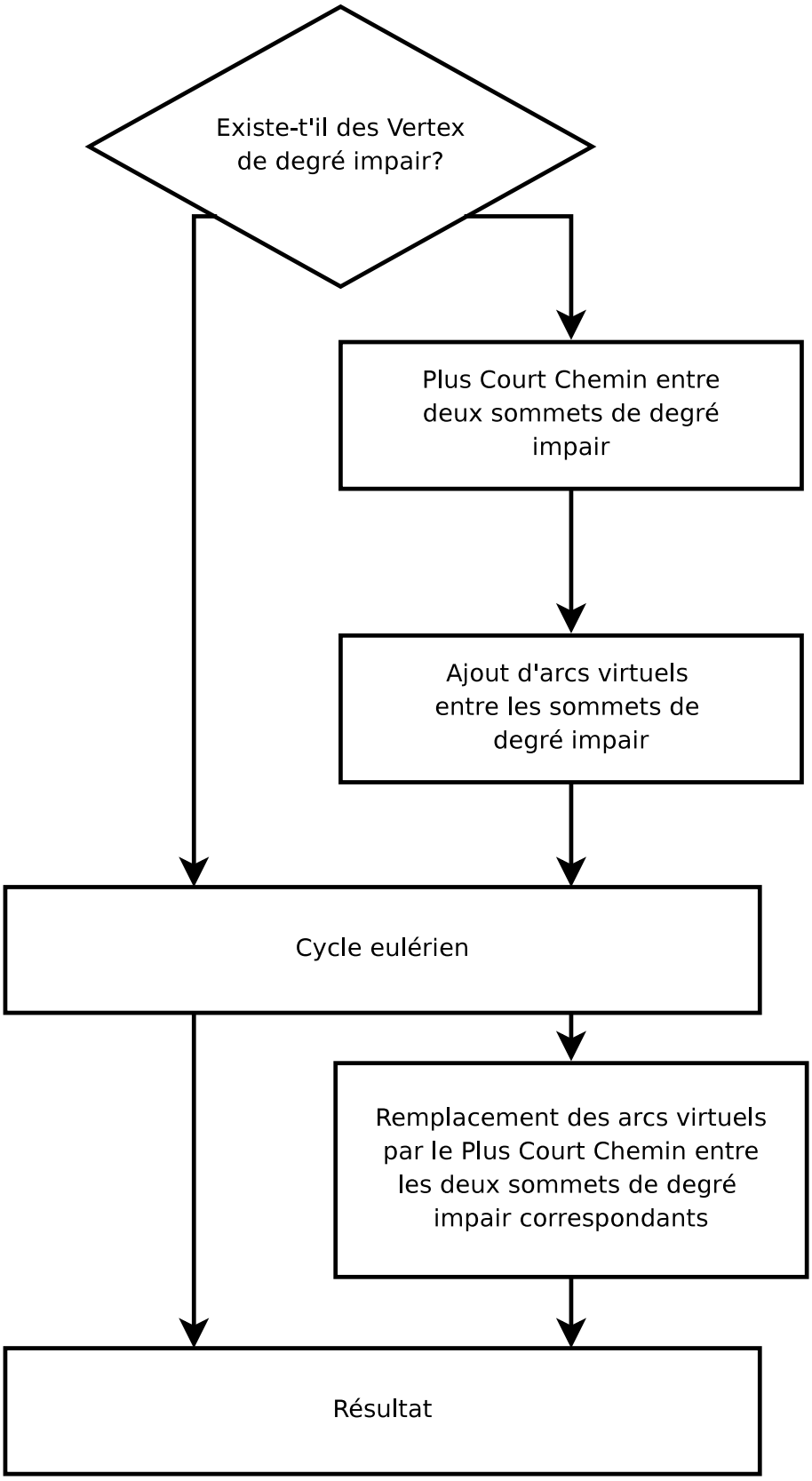
Un fois les fichiers obtenus, le défit est de transformer ces points sous la forme souhaitée : Vertex / Objets / Collection d'objet. Mais la contrainte est très forte car la Collection d'objet doit représenter le chemin que doit parcourir notre laser pour dessiner la forme 3D. L'enjeu est donc de parcourir un chemin de taille minimum, en passant par toutes les arrêtes et en coupant le minimum de fois le laser (le temps de coupure du laser étant long).

3.2.1 Parcours eulérien et Plus court chemin

Ce problème peut être traité selon la théorie des graphes en considérant que chaque objet est un graphe non vide, connexe, antireflexif, symétrique, et non orienté. On peut alors trouver dans ce graphe un cycle eulérien. On simplifiera volontairement le problème en cherchant un cycle eulérien non nécessairement minimum. Ce problème peut s'apparenter au problème du postier chinois.

Le problème se décompose de la manière suivante :
- Tous les sommets du graphe sont de degré pair : dans ce cas, il suffit de trouver le cycle eulérien
- Certains sommets sont de degré impair : dans ce cas, il faut relier virtuellement les sommets de degré impair les plus proches (création de nouveaux arcs), puis trouver le cycle eulérien, pour finalement remplacer les nouveaux arcs par un plus court chemin entre les deux sommets.

On résume le problème sous forme d'un schéma bloc.


Existe-t'il des Vertex de degré impair?
Plus Court Chemin entre deux sommets de degré impair
Ajout d'arcs virtuels entre les sommets de degré impair
Cycle eulérien
Remplacement des arcs virtuels par le Plus Court Chemin entre les deux sommets de degré impair correspondants
Résultat

PARCOURS EULERIEN

Algorithme. Voici l'algorithme adopter pour le calcul les cycles eulériens.

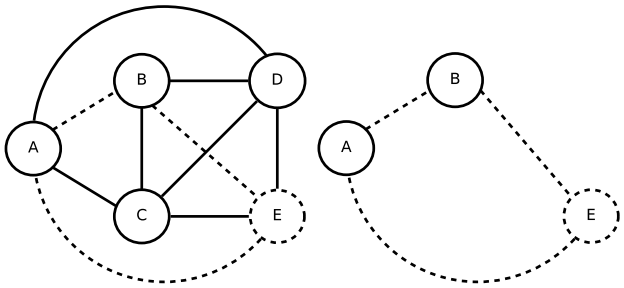
Algorithme 1 Cycle Eulérien

DONNÉES: E, Γ(non vide, connexe, antireflexif, symétrique, représentant un graphe non orienté)
RÉSULTAT: c

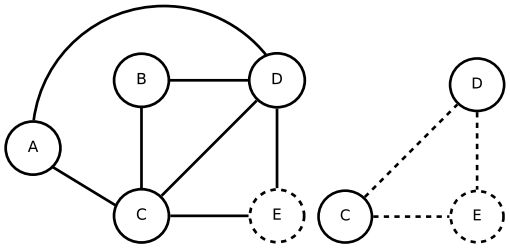
$z = 1$;
pour tout $x \in E$ **faire**
  $d(x) = |\Gamma(x)|$;
  si $d(x) \neq 0$ **alors**
    $z = x$;
  finsi
fin pour
pour tout $x \in E$ **faire**
  si $d(x)$ est impair **alors**
    $c = ()$;
    retourner
  finsi
fin pour
$c = (z); i = 1$;
tant que $d(z) > 0$ **faire**
  $x = z; c_1 = z$;
  répéter
    Choisir $y \in \Gamma(x)$
    $\Gamma(x) = \Gamma(x) \backslash \{y\}; \Gamma(y) = \Gamma(y) \backslash \{x\}; d(x) = d(x) - 1; d(y) = d(y) - 1$;
    $c_1 = c_1 + y; x = y$;
  jusqu'à $x = z$;
  Concaténer c_1 à c en i ;
  tant que $d(z) = 0$ et $i < |c|$ **faire**
    $i = i + 1; z = c[i]$;
  fin tantque
fin tantque

Sources : cours ESIEE PARIS, M. COUPRIE

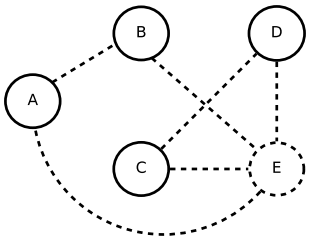
Explication de l'algorithme. Soit $G = (E, nonT)$ un graphe non oriente dont tous les sommets sont de degré pair. Si ce graphe est non vide, connexe, antireflexif, symétrique, et non orienté, on peut trouver un cycle eulérien. Dans notre exemple $E = a, b, c, d, e, f$, et on désigne arbitrairement e comme sommet initial. L'algorithme trouve alors un cycle dans E qui peut être par exemple $e \rightarrow a \rightarrow b \rightarrow e$.



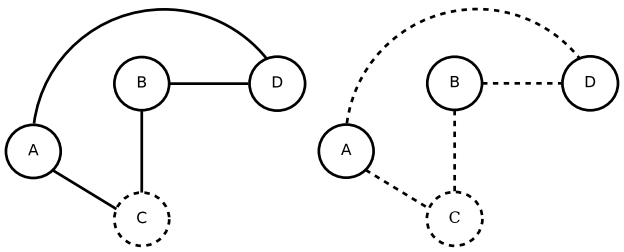
Comme le degré de e est de 2, il existe encore un cycle de e à e. On réitère donc l'opération précédente. Le second cycle est alors (par exemple) $e \rightarrow c \rightarrow d \rightarrow e$



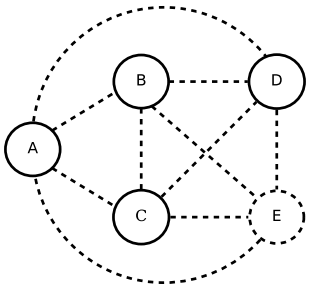
On concatène alors les deux cycles obtenus. Le cycle résultant est $e \rightarrow a \rightarrow b \rightarrow e \rightarrow c \rightarrow d \rightarrow e$.



Comme le degré de e est nul, on vérifie s'il existe des sommets de degré non nul parmis les sommets du cycle. On observe alors que le degré de c (par exmple) est de 2. On construit un nouveau cycle à partir de c qui est : $c \rightarrow b \rightarrow d \rightarrow a \rightarrow c$



De nouveau, on fusionne les cycles pour obtenir le cycle final : $e \to a \to b \to e \to c \to b \to d \to a \to c \to d \to e$ qui est le graphe G. Dans le cas où il resterait des sommets de degré non nul, on réitèrerait l'opération, mais ce n'est pas le cas ici.



PLUS COURT CHEMIN

Algorithme. Pour le calcul du plus court chemin, nous avons choisit d'utiliser l'algorithme de DIJKSTRA car le réseau est à longueurs positives.

Algorithme 2 Algorithme du Plus Court Chemin - DIJKSTRA

DONNÉES: $E, \Gamma, l, i \in E$
RÉSULTAT: π, S
  pour tout $x \in E \backslash \{i\}$ **faire**
    $\pi(x) = \infty$
  fin pour
  tant que $k < n$ et $\pi(x_k) < \infty$ **faire**
    pour tout $y \in \Gamma(x_k)$ tel que $y \notin S$ **faire**
      $\pi(y) = min[\pi(y), \pi(x_k) + l(x_k, y)]$
    fin pour
    Extraire un sommet $x \notin S$ tel que $\pi(x) = min[\pi(y), y \notin S]$
    $k = k + 1, x_k = x, S = S \cup \{x_k\}$
  fin tantque

Sources : cours ESIEE PARIS, M. COUPRIE

Explication de l'algorithme. L'algorithme de DIJKSTRA calcul le plus court chemin d'une racine i à un sommet $s \in E$. Cet algorithme présente l'avantage d'un temps de calcul en $O(n^2)$ voire en $0(nlog(n)+m)$ (algorithme de BELLMAN en $O(n(n+m))$). Nous rapellons que l'utilisation de cet algorithme a pour but d'établir des liens entre les sommets de notre objet 3D dont le degré est impair. On utilisera donc DIJKSTRA avec une racine i égale à un sommet du graphe dont le degré est impair. En fin d'algorithme, on compare les

distances entre i et les autres sommets de degré impair et on relie virtuellement ces deux sommets.

Après l'exécution de DIJKSTRA pour tous les sommets de degré impair, on cherche le cycle eulérien comme précédemment définit.

Enfin, on concatène le résultat de DIJKSTRA et d'EULER pour obtenir la solution.

(REMARQUE :) On peut prouver que dans notre cas, le nombre de sommet à degré impair sera toujours pair. On pourra donc associer les sommets de degré impair deux à deux.

(REMARQUE :) La solution adoptée n'est pas optimale, mais permet d'obtenir une solution satifaisante, en terme de temps de calcul (pour le PC) et de nombre de sommets (pour l'ARM7).

3.2.2 Exportation 3D sur microcontrôleur

La figure obtenue est exportée sur le microcontrôleur pour être compilée et exécutée. Faute de temps, nous n'avons pas pu adopter une solution permettant d'écrire l'information directement en mémoire du processeur.