Un jeu de tir LASER
ou comment mélanger le traitement de signal, l'électronique et l'assembleur

T. Rocacher, V. Mahout, P.-E. Hladik
thierry.rocacher@insa-toulouse.fr, vincent.mahout@insa-toulouse.fr
Département de Génie Électrique & Informatique INSA,
135 Av de Rangueil, 31077 Toulouse cedex 4

Résumé :

Cet article présente une maquette pédagogique qui sert de support aux enseignements de troisième année de la filière MIC (Modélisation Informatique et Communication), option Informatique et Réseaux. Cette maquette ludique permet d'illustrer sous la forme d'un bureau d'étude les domaines du traitement du signal, de l'électronique analogique, de la programmation en langage d'assemblage (ARM Cortex-M3). Elle sert à nouveau en quatrième année lors de la formation sur les périphériques de microcontrôleur (STM32F103). Il s'agit d'un jeu composé de plusieurs pistolets LASER et d'une cible électronique. But du jeu : plusieurs joueurs tirent simultanément sur une cible. La cible doit indiquer le nombre de fois où chaque tireur a fait mouche. Pour ajouter à la difficulté, un son doit être émis à chaque « impact »

Après une description du dispositif et du principe de fonctionnement, la structure des pistolets est détaillée ainsi que celle de la cible. Nous présentons ensuite le contenu pédagogique, en commençant par l'aspect traitement du signal, poursuivant avec l'électronique de traitement du capteur, et enfin en détaillant la programmation assembleur.

Mots clés : DFT, Transmission LASER, langage d'assemblage, format fractionnaire.

1. PRINCIPE, CONSTITUTION DU SYSTEME

Le jeu compte six pistolets en tout. Chacun d'eux est en réalité un jouet un peu amélioré puisqu'il embarque une diode LASER. La cible inclut une photodiode, l'électronique de traitement, un micro-contrôleur, un amplificateur BF et un haut-parleur.

Les figures 1 et 2 fournissent les schémas fonctionnels d'un pistolet et de la cible.

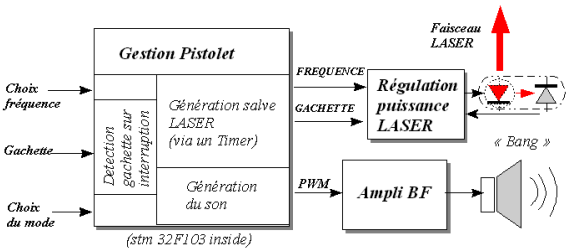

***Fig. 1**: Schéma fonctionnel d'un pistolet*

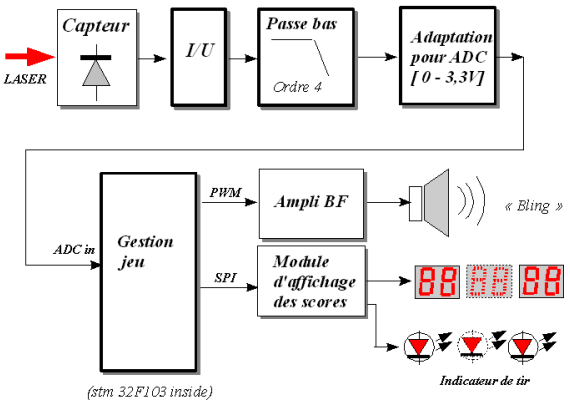

***Fig. 2**: Schéma fonctionnel de la cible*

Le pistolet est bâti autour d'un µcontrôleur de type STM32F103RB. Lorsque le tireur agît sur la gâchette, une salve LASER est émise durant 100ms. La figure 3 donne l'allure du courant dans la diode :

Courant dans la diode LASER

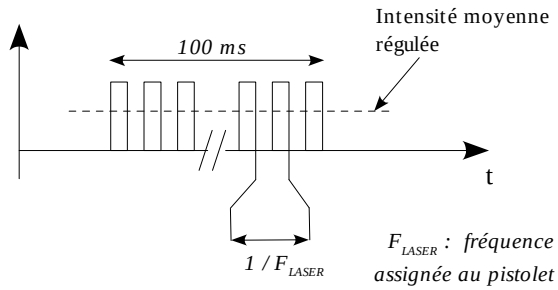

***Fig. 3**: forme d'onde de l'émission LASER*

La fréquence est choisie parmi six possibles. C'est la « signature du pistolet ». A noter que le pistolet est bridé pour ne déclencher que deux salves par seconde.

Lors de chaque émission LASER, le bruit de détonation du pistolet, emprunté au film « *les tontons flingueurs* » est produit. Cette partie n'est pas décrite dans l'article.

Enfin, un bouton « Mode » permet de couper le son. Très utile en TP...

La cible est l'objet principal de travail des étudiants (les pistolets sont donnés). Les fonctions de la figure 2 dont l'encadré est épais sont à concevoir et à réaliser par les étudiants (câblage, programmation). La difficulté majeure de l'application consiste à identifier les joueurs qui ont

Article publié par EDP Sciences et disponible sur le site http://www.j3ea.org ou http://dx.doi.org/10.1051/j3ea/2013012

touché la cible en fonction des signaux reçus. La solution que nous avons retenue est la ***T***ransformée de ***F***ourier ***D***iscrète. D'une part elle est tout à fait indiquée pour ce genre d'application, d'autre part, elle est extrêmement utilisée de nos jours en télécommunication (OFDM par exemple). C'est donc l'occasion pour nos étudiants non seulement de la voir de manière théorique, mais aussi de la programmer... en assembleur.

2. DESCRIPTION DES PISTOLETS

La figure 4 montre un pistolet démonté.

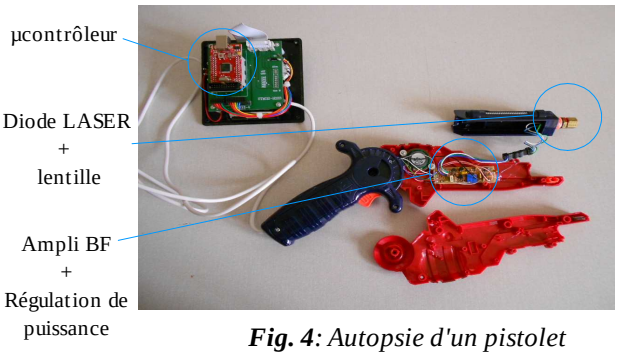

Fig. 4*: Autopsie d'un pistolet*

Il y a beaucoup à dire uniquement sur les pistolets. Nous avons fait le choix de ne détailler que la partie qui nous semble la plus délicate à réaliser, la régulation de la puissance du LASER.

La régulation de puissance (voir figure 5):

Les signaux *FREQUENCE* et *GACHETTE* sont générés par le microcontrôleur. Le signal *FREQUENCE* est un signal carré de fréquence fixe.

Dès l'action sur la gâchette mécanique du pistolet, le signal *GACHETTE* passe à '1' pendant 100ms, puis retombe à '0'.

- *On suppose que T3 est bloqué*

Lorsque le signal *GACHETTE* est au repos (0V), le transistor T1 est bloqué, la diode LASER est éteinte. La photodiode intégrée en regard du LASER est dans l'obscurité. Son courant inverse, I_R est nul. Le transistor T2 est donc bloqué aussi.

A l'inverse, quand le signal *GACHETTE* est à l'état haut, le condensateur C1 se charge à travers R2, ce qui met en conduction T1. Un courant I_D traverse la diode LASER qui s'éclaire et qui génère un courant I_R dans la photodiode par couplage optique. Ce courant fait augmenter le potentiel de base de T2, jusqu'à atteindre 1,2V environ (V_{D1} + V_{BE2}). La mise en conduction de T2 provoque une diminution du potentiel de base de T1 qui réduit alors l'intensité I_D du LASER. La boucle ainsi décrite succinctement permet une régulation de puissance du LASER. Le réglage de ce point de repos se fait par action sur le potentiomètre ajustable P0. En effet, comme le potentiel de base de T2 est à peu près constant (1,2V), le réglage de P0 agit sur le courant I_R donc sur le courant de la diode LASER via le couplage optique LASER-photodiode.

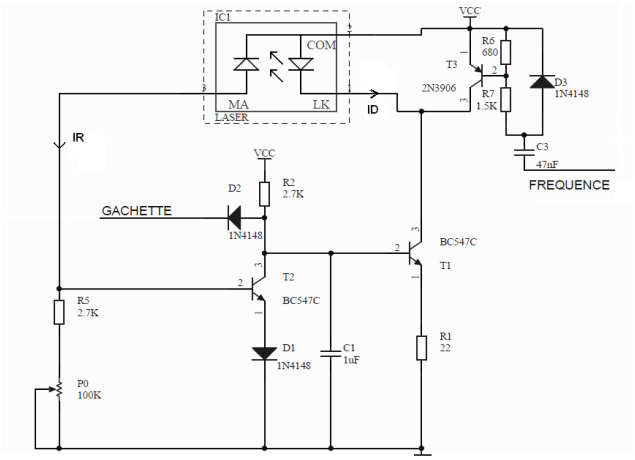

Fig. 5*: La régulation LASER*

- *On suppose que T3 est passant*

Si T3 est saturé, alors le courant I_D est dévié dans T3 (LASER éteint). Le signal *FREQUENCE* (figure 3) qui est un signal carré émis par le microcontrôleur provoque donc le découpage du faisceau LASER à la fréquence fixe.

3. DETAIL DE L'ANALYSE PAR DFT

C'est le coeur du dispositif. Nous allons aborder dans cette partie le choix des fréquences que nous utilisons pour les pistolets, ainsi que le paramétrage adéquat de la DFT à mettre en place, les deux aspects étant complètement liés.

3.1. Choix des fréquences : premières contraintes

Comme nous venons de le voir, chaque pistolet émet non pas une sinusoïde mais un signal carré. Trois problèmes se posent alors :

– Le spectre est trop large à cause des harmoniques impairs intrinsèques à la forme carrée, et va engendrer un repliement de spectre qui va brouiller la DFT suite à l'échantillonnage de l'ADC.

– Le choix de fréquences est délicat. L'ensemble des six fréquences possibles doit être à bande « assez » étroite. En effet, la régulation que nous avons vue exige un découpage relativement élevé en fréquence, afin de ne pas gêner la régulation moyenne du courant.

Premières conclusions

Ces contraintes physiques incontournables nous amènent déjà à deux conclusions : les six fréquences seront assez élevées, tout en restant dans la bande passante de l'ensemble LASER – photodiode de réception. On visera un centrage aux environs de 100 kHz. De plus, l'ADC de réception devra être précédé d'un filtre anti-repliement efficace (un ordre 1 ne sera très probablement pas suffisant). Il aura une bande passante élevée, ce qui rendra l'utilisation d'AOP possibles, mais délicate.

3.2. Paramétrage de la DFT

Afin de déterminer ses paramètres (nombre de points, fréquence d'échantillonnage) il convient de bien connaître les aspects échantillonnage et surtout fenêtrage propres à la DFT. C'est l'objectif du document [ROC] qui présente ces aspects, et qui ne traite que de la fenêtre rectangulaire. Il est donné aux étudiants comme support de cours. On pourra trouver des informations plus larges et plus complètes ([BAU]).

Rappelons la transformée de Fourier discrète :

$$X(k) = \frac{1}{M} . \sum_{n=0}^{n=M-1} x(n) . e^{-j\frac{2.\pi.k.n}{M}} \quad (1)$$

Intéressons-nous aux signaux intermédiaires *e(t)*, signal continu à traiter, $e_T(t)$, signal fenêtré, et enfin $e_T^*(t)$ signal fenêtré et échantillonné, nous obtenons les spectres respectifs de la figure 6. Précisons que *T* est la durée d'ouverture de la fenêtre.

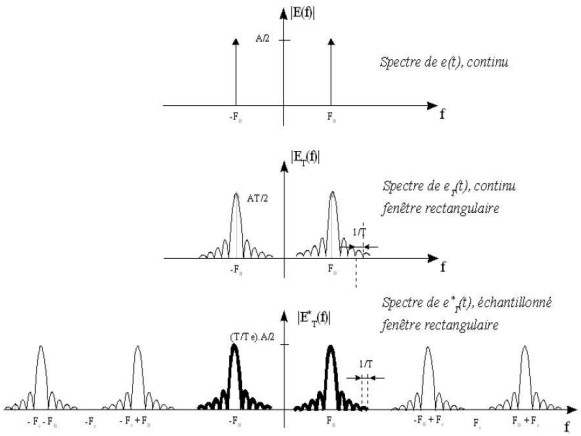

Fig. 6*: Spectre des signaux mis en jeu au cours de l'acquisition*

Cette figure ne montre pas encore l'échantillonnage fréquentiel (donc la suite *X(k)*) mais elle éclaire en amont, sur l'effet du fenêtrage rectangulaire : Toute impulsion de Dirac dans le spectre continu d'origine est remplacée par un sinus cardinal centré sur l'impulsion.

Sachant d'une part que les passages à 0 du sinus cardinal sont des multiples de 1/*T*, et que par définition les *X(k)* sont les échantillons du spectre pris tous les 1/*T*, on en déduit :

Si l'on veut avoir six canaux qui ne se perturbent pas, c'est à dire un tir de pistolet qui donne une raie unique, chaque fréquence doit s'écrire :

$$\boxed{F_k = k.\Delta F = k.\frac{1}{T}} \quad (2)$$

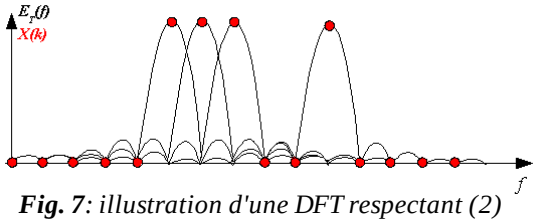

Fig. 7*: illustration d'une DFT respectant (2) pour 4 fréquences différentes.*

3.3. Choix des fréquences : seconde contrainte

Le dernier problème qui subsiste encore, est celui lié à la génèse des fréquences au niveau des pistolets. Le principe est basé sur l'utilisation d'un timer qui est à l'origine de l'un signal PWM à 50%, fixe. La finesse de résolution temporelle est liée à la période d'horloge qui commande le Timer du STM32F103RB : T_{CK} = 1 / 72MHz.

La période du signal LASER est donné par

$$T_{LASER} = N.T_{ck} \quad \text{soit} \quad F_{LASER} = \frac{F_{ck}}{N} \quad , N \text{ entier (3)}$$

Or, nous venons de voir que la condition (2) doit être respectée. Ce qui veut dire :

$F_{LASER} = \frac{F_{ck}}{N} = k.\Delta F$ (4). L'objectif est donc de trouver une valeur ΔF ainsi que six couples de valeurs {*N*,*k*} qui satisfont (4).

Ces considérations ainsi que les premières conclusions du paragraphe 3.1 (F_1 à F_6 formant une bande « assez étroite » centrée sur 100 kHz environ) nous ont amené à :

***T* = 200µs**, soit ***ΔF* = 5kHz.**
F_1 = 85kHz, F_2 = 90kHz, F_3 = 95kHz,
F_4 = 100kHz, F_5 = 115kHz, F_6 = 120kHz.

Sur les six fréquences ainsi imposées, trois sont exactes car satisfont l'équation (4), trois sont arrondies à 16Hz près dans le pire cas, ce qui est acceptable

Enfin, il faut $Fe > 2.F_6$ = 240kHz. De plus si N (nombre de points) est une puissance de 2, on pourra utiliser la FFT de MATLAB, on propose donc :

***Fe* = 320 kHz,** ce qui donne ***N* = 64**.

4. EXPLOITATION PEDAGOGIQUE

Ce travail est proposé aux étudiants de 3ème année du cursus MIC (Modélisation, Informatique et Communication) souhaitant intégrer en 4e année la filière Informatique & Réseau.

Le travail demandé est réalisé au cours d'un bureau d'étude de 13 séances de 2h45. Les six premières séances sont consacrées aux aspects électronique et traitement du signal. Durant cette première phase, les étudiants sont en contact avec une platine d'essais et l'oscilloscope pour réaliser le traitement du capteur, mais également à Matlab pour assimiler les bases théoriques de la DFT.

Dans une seconde série de six séances, ils devront gérer un son à produire et coder leur DFT en langage d'assemblage et en virgule fixe sur un micro-contrôleur.

Enfin, deux séances sont laissées libres pour que les étudiants finalisent le projet. L'expérience de l'an passé nous montre que pour certains binômes, ces deux séances serviront uniquement à faire le travail spécifiquement décrit (réception, discrimination DFT du pistolet émetteur), pour d'autres plus avancés, elles permettront de finaliser le projet en proposant un scénario de jeu intégrant les six pistolets.

4.1. Le traitement de signal

Le traitement de signal consiste à opérer la DFT sur un bloc d'échantillons. Dans un premier temps (5h30), nous insistons sur les aspects échantillonnage et fenêtrage inhérents à la DFT par un travail progressif sous Simulink accompagné du cours sous format *pdf*. A l'issue de cette étape, les étudiants doivent être capables de proposer et de justifier un choix de fréquences des pistolets ainsi que le réglage de la DFT associé. Ayant acquis ce minimum de maîtrise, le but sera de coder l'algorithme en langage d'assemblage, en quelques 12h00 de séances. Il est donc exclu de traiter le problème par une FFT, trop longue et complexe à mettre en place. Le choix s'est donc porté tout naturellement vers le calcul plus simple de six points de la DFT, au niveau des six fréquences possibles des pistolets.

4.2. L' ÉLECTRONIQUE DE TRAITEMENT

Les étudiants ont à concevoir cette électronique. Le traitement du capteur est relativement simple, il s'agit d'un simple convertisseur courant – tension à AOP.

Plus compliqué est le filtre anti-repliement. Il doit laisser passer une bande allant jusqu'à 120kHz et doit atténuer de manière conséquente à 160kHz (*Fe*/2). De plus il doit être simple à réaliser pour des débutants en électronique.

La solution proposée est celle d'un filtre de Chebychev d'ordre 4. On cascadera donc deux filtres résonants d'ordre 2. Comme les fréquences sont élevées, nous avons rejeté une solution type structure de Sallen-key, pour préférer deux filtres RLC série interfacés par un AOP en suiveur. C'est aussi l' occasion de revisiter ce grand classique des systèmes du second ordre.

La figure 8 est la photographie d'une réalisation étudiante.

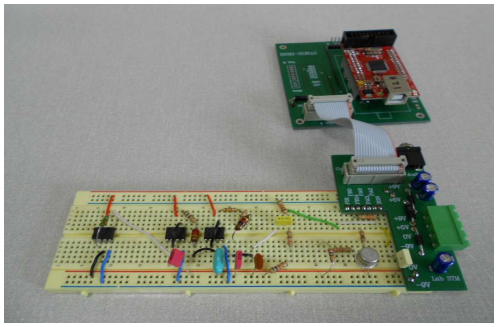

***Fig. 8**: Réalisation de l'électronique*

4.3. LE TRAVAIL SUR MICRO-CONTROLEUR

Comme ce bureau d'étude s'adresse à de futurs étudiants de la filière Informatique et Réseaux l'aspect codage, contrairement aux aspects électronique et signal, coïncide donc généralement mieux à leurs compétences et à leurs motivations. Cependant, par rapport à leur acquis précédents, ils vont être confrontés à trois nouveautés qui vont engendrer chez eux questions et réflexions :

- la programmation en développement croisé ,
- la programmation en langage d'assemblage ,
- le codage des nombres en virgule fixe.

4.3.1. Le contexte en développement croisé

D'un point de vue strictement technique les cibles sont équipées de carte prototype Olimex H103 [OLI] et le logiciel de développement sous Windows est Keil µVison3 [KEI].

Par développement croisé, on entend le fait de développer un programme sur station (PC), programme qui en réalité sera exécuté sur un tout autre processeur.

Le concept de développement croisé est une notion pédagogiquement intéressante pour des élèves-ingénieurs. Il ne s'agit en effet plus pour eux

d'écrire une suite d'instructions qui affichera des résultats plus ou moins compliqués à obtenir dans l'environnement dans lequel ils codent. Ils doivent dans le cadre de cet exercice écrire un programme pour un système qui n'est doté ni d'un écran ni d'un clavier. La mise au point des algorithmes doit donc faire appel à des techniques de débogage (non décrites dans cet article) qui mettent à mal leurs habitudes de test. Nous insistons notamment pour que les étudiants apprennent à tester leur code, par les moyens appropriés, mais sans ajouter ni retirer de la version de l'application qui sera finalement embarquée.

Le processus étant physique, il est nécessaire de concevoir les routines logicielles qui permettent d'attaquer les unités périphériques du µcontrôleur. Ces unités sont pour l'essentiel :

- un timer pour lancer les conversions à 320khz,
- un convertisseur analogique / numérique programmé pour ranger les 64 échantillons directement en mémoire (DMA),
- une PWM pour jouer le son sur le haut-parleur,
- des entrées sorties binaires pour les LEDs et interrupteurs,
- une liaison série pour gérer les différents afficheurs.

Toutes les routines permettant d'utiliser ces différentes ressources sont données (en code objet) aux étudiants. En effet, la programmation des unités périphériques d'un microcontrôleur fait partie d'une UF de 4e année et ne fait donc clairement pas partie des objectifs pédagogiques de ce BE.

4.3.2. La programmation en langage d'assemblage

Lorsque les étudiants commencent à coder l'application, ils, ont normalement acquis les compétences pour le traitement d'un signal par DFT. Ils ont également suivi un enseignement (10h00 de cours et 5h00 de TD) pour leur expliquer les bases et les principes de la programmation en langage d'assemblage.

Le microcontrôleur de la carte est un STM32F103RB. Il s'agit d'un processeur qui, comme son nom peut le laisser deviner est un processeur 32 bits conçu par ST Microelectronics.

Ce processeur est bâti autour d'un architecture ARM Cortex-M3. C'est donc le langage d'assemblage de ce cœur qui est vue en cours [MAH] et qui est mis en œuvre par l'environnement Keil. Cet outil recourt à l'assembleur RealView d'ARM pour façonner l'exécutable. Il est à noter que pour les besoins de ce BE la version gratuite et limitée de cet environnement est largement suffisante. Cependant l'utilisation d'un assembleur de type gnu/gcc est également possible puisque l'environnement Keil l'intègre.

La partie codage est elle-même décomposée en trois parties.

La première est, pédagogiquement parlant, extrêmement guidée. Elle a pour finalité de « jouer » un son sur le haut-parleur. Concrètement cela revient à déclarer le son comme un tableau de données (un script Matlab fourni leur permet d'engendrer ces données à partir du fichier .wav de leur choix, à condition qu'il ne soit pas trop conséquent) et à le sortir en faisant appel à la routine liée à la PWM. Ce premier objectif est modeste en terme d'algorithmique, mais il est suffisamment contraint pour obliger à mettre en œuvre les éléments clé de la programmation en langage d'assemblage : les principales boucles algorithmiques (if...then, while, for), la gestion simple de tableaux et de pointeurs, l'appel à des procédures avec leur passage d'arguments.

La seconde partie concerne la codage de la DFT en elle-même (voir le paragraphe suivant). Cette étape est moins guidée (seule la structuration globale en différentes fonctions est donnée, les algorithmes sont donc à produire).

Enfin dans une dernière phase, les étudiants sont amenés à intégrer l'ensemble de leur routine dans le projet final. Au cours de cette étape, nous les incitons à faire l'intégration en langage C. Cette option les oblige à réfléchir sur l'interface entre ces deux langages et à s'interroger sur la façon dont travaille un compilateur.

4.3.3. Le codage de le DFT en virgule fixe

le STM32 ne possède pas d'unité à virgule flottante (ALU uniquement sur des entiers 32 bits) et nous interdisons le recours à des bibliothèques mathématiques en virgule flottante.

Nul choix n'est donc laissé et le codage doit donc passer par l'appropriation du codage des nombres réels en format fractionnaire (virgule fixe).

Si pour un programmeur il est assez aisé de concevoir le codage de :

$$X(k) = \sum_{n=0}^{M-1} x(n)\left(\cos\left(2\pi\frac{nk}{M}\right) - j\sin\left(2\pi\frac{nk}{M}\right)\right)$$

il devient tout de suite plus compliqué d'imaginer la routine lorsque l'on ne dispose pas des fonctions trigonométriques, et que le signal *x(n)* est un entier 16 bits avec uniquement 12 bits significatifs alignés à gauche (format issu de l'ADC).

Le codage passe donc par la tabulation (en format 1.15) des 64 points nécessaires pour décrire les fonctions trigonométriques.

Une des premières demandes de cette étape, qui leur est imposée afin de leur permettre de s'aguerrir sur ce format particulier, est de vérifier avec leurs

valeurs de sinus et de cosinus tabulées que $\cos^2(nk)+\sin^2(nk)=1$ pour un *nk* quelconque.

Ils sont ensuite guidés dans leur réflexion pour déterminer les calculs et les renormalisations nécessaires pour que le résultat (la norme au carré des *X(k)*) soit au final un entier 32 bits codé en 8.24 (8 bits de partie entière et 24 bits de partie décimale).

La mise au point de ces routines est particulièrement délicate. Pour s'en convaincre, sauf à avoir un FPGA à la place du cervelet, il suffit d'essayer de « lire » dans le texte la valeur hexadécimale 16 bits 0x04B0 en sachant qu'elle correspond à un réel en 8.8. La bonne réponse est 4,6875 !

Pour faciliter cette mise en œuvre nous avons écrit à l'usage des étudiants des macros de l'environnement de développement qui permettent de faire ces conversions simplement. Ainsi ils peuvent à tout moment (en déclenchant l'appel à ces macros de débogage sur des points d'arrêt conditionnés par exemple) vérifier la cohérence et le bon déroulement de leur calcul.

Cette phase n'est certainement pas la plus aisée à mettre au point mais elle est réellement à la croisée des chemins (codage des nombres, langage d'assemblage, traitement du signal) et concrétise bien l'interdisciplinarité que nous avons cherchée à valoriser à travers ce bureau d'étude.

5. CONCLUSION

Ce BE a été relativement long à mettre en place. Il a nécessité beaucoup de travail de mise au point, et a considérablement occupé l'équipe technique du GEI. En effet, il doit pouvoir être dispensé à deux groupes de TP (2x12 étudiants) en même temps, ce qui a conduit à la production de séries de maquettes importantes.

***Fig. 9**: La cible*

L'aspect ludique du dispositif a sans doute contribué à garder une motivation suffisante des étudiants, qui leur a permis de s'initier à des domaines à la marge de l'informatique pure, sur une période de presque un semestre entier.

6. REMERCIEMENTS

Nous tenons à remercier particulièrement Daniéla DRAGOMIRESCU ainsi que Jean-Louis NOULLET qui ont été les précurseurs, 10 ans plus tôt, de ce bureau d'étude dont nous avons repris intégralement l'électronique de régulation très astucieuse du LASER. Remercions aussi l'équipe pédagogique pro-active dans ce projet, D.LE BOTLAN, M.AIME, G.AURIOL, P.ACCO, ainsi que l'ensemble de l'équipe technique du GEI, J.MARTIN, J.PEREZ, F.NOUGAROLLES, J,-F. RIBAS, G.RAYSSAC, pour la réalisation de l'ensemble des maquettes et S.DIMERCURIO, pour son aide précieuse sur le STM32F103RB.

7. BIBLIOGRAPHIE

[OLI] : site du concepteur des cartes prototype : *www.olimex.com/Products/ARM/ST/STM32-H103/*

[KEI] : site de l'environnement de développement (lié à ARM) : www.keil.com

[MAH] : MAHOUT Vincent , *Programmation en langage d'assemblage : ARM Cortex-M3* , Hermes N°ISBN 978-2-7462-3211-2, 2011

[STM]: page d'accueil de STMicroelectronics sur le STM32 : http://www.st.com/internet/mcu/class/1734.jsp

[BAU]: G. BAUDOIN et J.-F. BERCHER , « *Transformée de Fourier Discrète* » , ESIEE ,sur : www.esiee.fr/~bercherj/New/polys/poly_tfd.pdf, nov 2001

[ROC]: ROCACHER Thierry, « *La transformée de Fourier dicrète* », INSA Toulouse, sur :

http://www.lesia.insa-toulouse.fr/~acco/acco_wiki/doku.php?id=hard:cours_elec:accueil

[MOO] : Site pédagogique INSA de Toulouse, page du bureau d'étude :

http://moodle.insa-toulouse.fr/course/view.php?id=528