TRIEUSE DE PIECES

DOSSIER TECHNIQUE

MATHERON Aimé

LAGARRIGUE Jérémy

MARRET Alexine

KOENIG Enola

SOMMAIRE

I. Présentation du projet :

1) Cahier des charges et schéma fonctionnel :

Depuis l'essor de l'Institut d'Optique Graduate School dans le classement des grandes écoles, le nombre de ses étudiants croit considérablement. Malheureusement, le nombre de machine à café ou de vendeur au Safran ne suit pas cette évolution et une queue est facilement formée à chaque pause.

L'objectif de ce projet est de réaliser une machine capable de trier des pièces et de les ranger séparément afin d'optimiser le temps passer à chercher sa monnaie.

Pour répondre aux besoins du client, la trieuse de pièces devra remplir les fonctions suivantes :

- Détecter la valeur d'une pièce
- Etre mécanisée pour pouvoir trier et ranger la pièce de manière autonome
- Posséder une structure solide assurant sa durée de vie et sa bonne utilisation

Le cahier des charges de cette trieuse de pièces, contenant les composants permettant de réaliser chaque fonction est donc le suivant :

Objectifs	Description	Composants/critères
F1 : identifier la valeur de la pièce	Peser la pièce à l'aide d'un système de balance pour en déduire sa valeur	Balance de bijoutier amateur Précision minimum : 0,05g
F2 : sélectionner la bonne boite correspondant à la pièce	Faire tourner le plateau supérieur de la machine pour positionner la sortie de la rampe au-dessus de la boîte correspondant à la pièce	Moteur pas à pas fixé sur l'axe vertical de la trieuse
F3 : pousser la pièce dans la rampe et la faire glisser	Pousser la pièce dans la rampe à l'aide d'un système de barre coulissante	Servomoteur standard sur lequel est fixée la tige pousseuse
F4 : illuminer la machine en fonction de la pièce triée	Allumer un ruban de LED d'une certaine couleur en fonction de la valeur de la pièce	Ruban de LED générant 8 couleurs différentes
F5 : structure solide de la machine	Structure solide contenant l'électronique du système : carte Nucléo, circuits électroniques, fils, servomoteur, moteur pas à pas, balance, boîtes de triage, rampe	Pièces de bois découpées à la découpe laser
F6 : communication entre les différentes parties de la machine	Programme informatique	Carte Nucléo Câbles de connexion

Figure 1 : Cahier des charges de la trieuse de pièces :

Dans l'ordre chronologique d'utilisation, la trieuse de pièce détecte une pièce puis envoie cette information au programme informatique. Celui-ci déclenche alors simultanément l'illumination

et la rotation de la machine vers la boite adéquate en fonction de la valeur de la pièce détectée. Une fois la rotation finie, la pièce est poussée dans la rampe, puis un court laps de temps plus tard, la trieuse revient dans sa position initiale, prête à trier une autre pièce. Le schéma fonctionnel de la trieuse de pièces est le suivant :

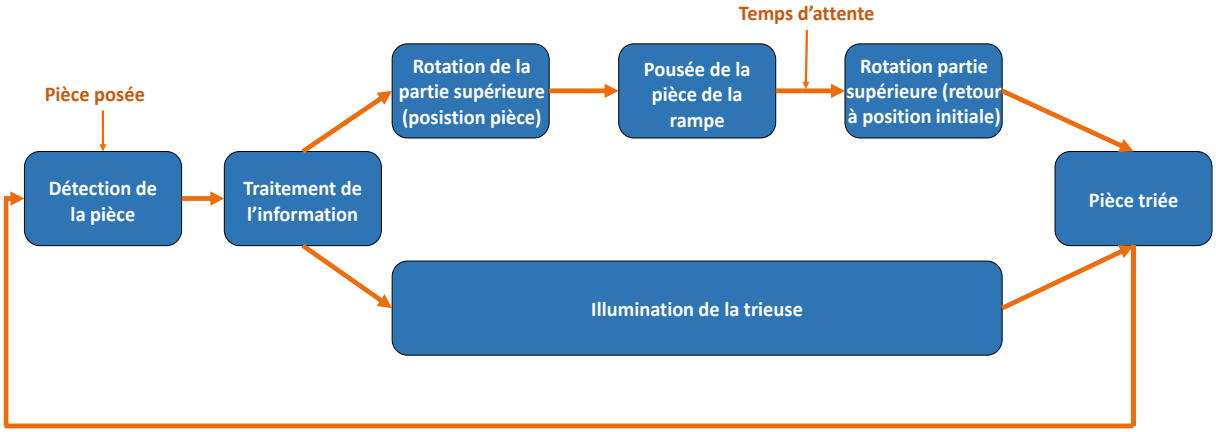


Figure 2 : Schéma fonctionnel de la trieuse de pièces :

2) Répartition des tâches et planning final :

Pour ce projet nous avons choisi de travailler par binôme pour la plupart des étapes de conception afin d'optimiser la réflexion et le temps de travail.

Ainsi, Aimé et Jérémy ont travaillé ensemble sur le mécanisme de rotation de la tige qui permettra de pousser la pièce dans la rampe pendant la première séance. Ils se sont ensuite concentrés pendant les six autres séances au système de détection et d'identification de la valeur de la pièce en récupérant le signal de sortie d'une balance.

Alexine et Enola se sont occupées du mécanisme de rotation global de la machine pendant les trois premières séances, puis elles se sont ensuite chargées d'illuminer la machine en fonction de la pièce détectée et à trier. Sur les deux séances suivantes elles se sont divisées pour plus d'efficacité afin de réaliser des tâches moins complexes : Alexine s'est donc occupée de la modélisation et de l'agencement de la structure finale de la trieuse tandis qu'Enola a réalisé le code global permettant de faire fonctionner tous les éléments ensemble. Sur la dernière séance elles ont découpé les pièces et monté la structure de la machine.

Nous avions établi un planning initial en répartissant les tâches en fonction de leur ordre de priorité et d'importance pour le fonctionnement de la trieuse. Cependant, ce planning n'a pas pu être respecté à la lettre car certaines étapes clés nous ont pris plus de temps que prévu. C'est le cas de la rotation de la plateforme avec le moteur pas à pas, mais surtout, du système d'identification de la valeur de la pièce qui a demandé beaucoup de travail. Finalement, Aimé et Jérémy se sont occupés de la fonction centrale de la machine qui a posé le plus de soucis pendant qu'Alexine et Enola ont réalisé l'ensemble des autres tâches pour avancer au mieux le projet. Le planning final des tâches est donc le suivant :

SEANCE 1

Aimé et Jérémy ont fait fonctionner un servomoteur standard pour qu'il réalise un aller-retour de 180° afin de pousser la pièce dans la rampe.
Alexine et Enola ont fait fonctionner un servomoteur continue pour faire tourner l'ensemble de la machine mais elles se sont rendu compte que le couple du moteur serait insuffisant elles ont donc décidé d'utiliser un moteur pas à pas.
Le groupe s'est fixé sur la structure de la trieuse.

SEANCE 2

Aimé et Jérémy ont commencé à étudier la balance pour détecter la valeur de la pièce
Alexine et Enola ont pris en main le moteur pas à pas et commencé à programmer sa rotation

SEANCE 3

Aimé et Jérémy ont créé un circuit avec amplificateur et filtre passe-bas pour réussir à détecter une tension correcte en présence d'une pièce sur la balance.
Alexine et Enola ont réussi à faire tourner le moteur pas à pas dans les deux sens pour un nombre de pas souhaité.

SEANCE 4

Aimé et Jérémy ont continué à résoudre les problèmes de détection de la pièce en supprimant davantage le bruit dans le signal de sortie.
Alexine et Enola ont étudié un ruban de LED trichrome et l'ont programmé pour qu'il fournisse huit couleurs différentes

SEANCE 5

Aimé et Jérémy essaient toujours d'améliorer la détection de différentes pièces sur la balance avec utilisation d'une carte Nucléo.
Alexine a modélisé la rampe sur Audio Desk fusion 3D
Enola a rassemblé les codes et créé un programme pour faire fonctionner tous les éléments ensemble.

SEANCE 6

Aimé et Jérémy ont réussi à détecter et identifier différentes pièces sur la balance.
Alexine rejoint par Aimé s'est occupé de la découpe laser des pièces de la structure.
Enola rejoint par Jérémy à optimiser le code de la trieuse pour y intégrer celui de détection de la pièce.

SEANCE 7

Jérémy a continué d'optimiser le code de détection de la valeur de la pièce.
Alexine a continué de découper le reste de la machine et Enola l'a rejointe pour continuer de la construire.
Aimé et Enola ont refait les montages électriques sur une plaquette intégrable dans la structure et l'ont testé.

II. Réalisation du prototype :

1) Détection de la valeur de la pièce :

Cette partie consiste à réaliser le montage permettant de détecter la valeur de la pièce. Pour ce faire, nous avons choisi d'utiliser une balance de précision 0.01g car le poids entre deux pièces consécutives était de cet ordre de grandeur. La balance électronique sortait une valeur digitale du poids de la pièce codée sur 16 broches, inutilisable dans notre projet. La solution choisie a alors été de venir directement prélever le signal à la sortie de la jauge de contrainte, pièce maitresse de la balance. En se déformant, la jauge de contrainte étire certaines résistances variables et crée à ses bornes une différence de potentiel mesurable. C'est ce signal que nous mesurons dans la suite pour détecter les différentes pièces. Le signal étant de qualité médiocre, nous l'avons traité afin de pouvoir l'exploiter. On détaille ci-dessous le schéma global du traitement du signal de sortie de la balance que l'on peut résumer avec le schéma fonctionnel suivant :

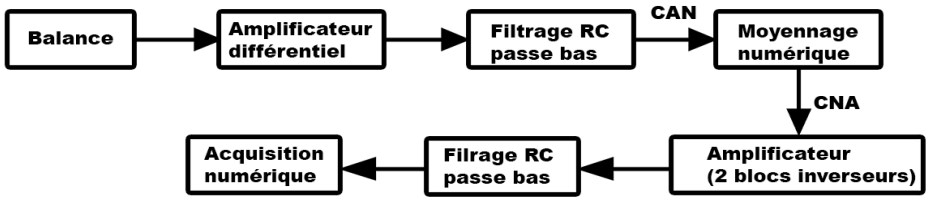


Figure 3 : Schéma fonctionnel de la détection de la valeur de la pièce

Comme indiqué, on récupère le signal faible et bruité, en sortie de la balance, que l'on différencie. Cette différenciation entre les 2 sorties de la balance permet d'obtenir une seule tension à amplifier. Le montage différentiel peut également amplifier la différence des entrées. Ainsi, on amplifie x 50 pour obtenir un signal plus sensible en sortie.

Malheureusement, on rajoute alors encore plus de bruit sur la sortie. Pour le réduire on filtre passe bas avec un circuit RC du premier ordre. Le signal ainsi amplifié et filtré est injecté dans la carte Nucléo.

On procède alors à un traitement numérique du signal via un programme détaillé par la suite. Cet algorithme va moyenner le signal sur un grand nombre de points ce qui diminue le bruit situé à haute fréquence.

On fait ensuite ressortir le signal moyenné de la Nucléo. On multiplie par 100 le signal de sortie. Pour cela on utilise des amplificateurs opérationnels placés en montage amplificateur inverseur. On décide de concaténer deux montages inverseurs de gain global x(-10), ainsi, le gain total en sortie est bien de (-10)*(-10)=100 ! Le choix d'avoir utilisé deux montages inverseurs x(-10) au lieu d'un seul montage non inverseur x100 peut être justifié par le fait que lorsqu'on cherche à obtenir un gain trop grand (au-delà de x80 pour un ordre de grandeur) dans un montage à amplificateur opérationnel, on voit apparaitre des comportements étranges du montage (sans doute des effets de seuil). Nous avons donc opté pour le choix de deux montages inverseurs.

Montage électrique

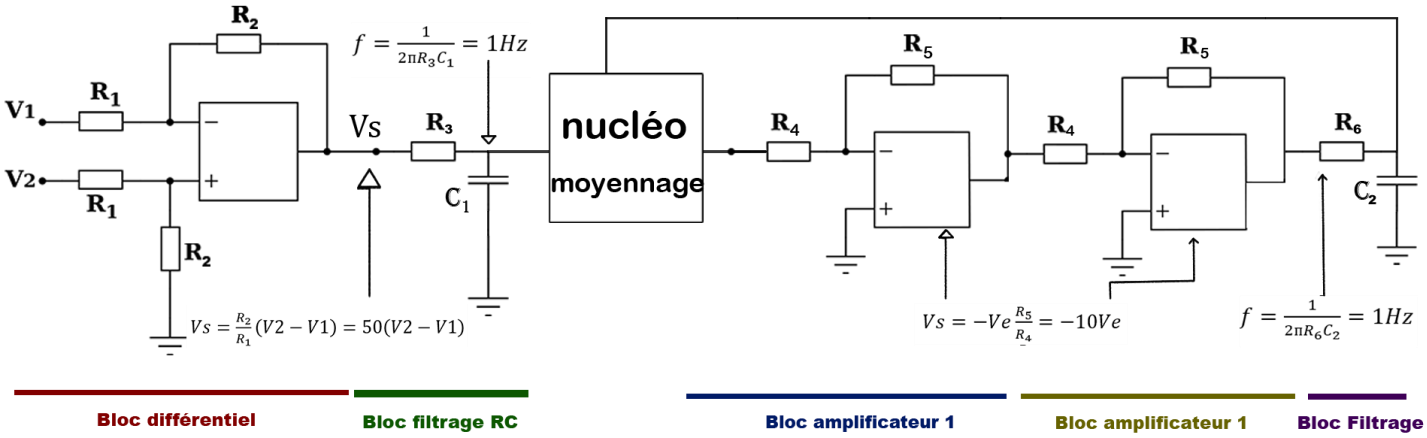


Figure 4 : Schéma électrique du circuit de détection de la valeur de la pièce :

Remarque : Pour les filtres RC on choisit $C_1 = 10nF$ et $R_3 \simeq 16\,m\Omega$ tel qu'on ait $f = \frac{1}{2\pi R_6 C_2} = 1Hz$.

Description de l'algorithme

Pour traiter le signal, nous le branchons en entrée de la Nucléo (entre A0 et la masse par exemple), elle nous reverra le signal de sortie traité entre la masse et PA_5. On crée une fonction *convert* qui moyenne le signal d'entrée sur 128 valeurs successives prises avec une fréquence d'échantillonnage de 100Hz.

On réalise un premier moyennage informatique du signal grâce à la fonction cette fonction en sortie du filtrage RC. Cela permet de diminuer davantage le bruit. Ce signal moyenné est ensuite amplifié par les blocs amplificateurs 1 et 2 puis renvoyé dans la Nucléo.

La valeur de la pièce étant croissante avec sa masse (donc avec la tension détectée), nous associons des plages de tension à une valeur précise de pièce (ex : 0.587 V < tension < 0.592 V ⟺ 1 euro). Le signal moyenné et amplifié envoyé dans la carte est plutôt stable et précis, toutefois il arrive que le signal empiète sur une plage inférieure ou supérieure à la plage théorique correspondant à la pièce. L'algorithme peut donc parfois renvoyer 1 euro alors que la pièce posée est 2 euros. Heureusement la valeur affichée est la bonne dans la majorité des cas, mais ce petit défaut doit être pris en compte. Un traitement statistique s'impose.

Ainsi, dans la fonction *DéterminerPieces* le programme fait tourner pendant 5 secondes la fonction *convert* et compte le nombre d'apparitions de chaque valeur de pièce et le nombre total de leurs apparitions. On calcule ensuite chaque fréquence d'apparition : $f = \frac{nb\_app}{nb\_tot}$. On décide que si une fréquence est supérieure à 0.7 alors elle donne la valeur de la pièce.

2) Motorisation de la structure :

Pour faire tourner la structure et positionner la rampe avec la boite correspondant à la pièce sélectionnée nous faisons tourner l'axe de la trieuse sur lequel est fixé le plateau supérieur de la machine grâce à un moteur pas à pas. Faire tourner la partie du haut plutôt que le plateau inférieur sur lequel repose les boites nous permet de nous affranchir de la variation de poids du plateau dû à l'augmentation du nombre de pièces triées et donc de la variation de couple nécessaire à la rotation de la structure qui l'empêcherait à long terme de tourner. Le moteur pas à pas est lui avantageux par rapport à un servomoteur continue car son couple est plus élevé et il est facilement contrôlable en angle, ce que l'on souhaite faire ici.

Le moteur pas à pas utilisé est un moteur bipolaire 220 pas, c'est-à-dire, constitué de deux bobines indépendamment contrôlables. La carte Nucléo seule ne fournit pas une tension suffisamment élevée pour alimenter le moteur. On utilise donc un convertisseur de puissance : un pont en H, directement intégré dans le composant L293D, pour pouvoir amplifier la tension d'alimentation du moteur et alimenter séparément chacune des bobines.

Schéma du branchement du moteur

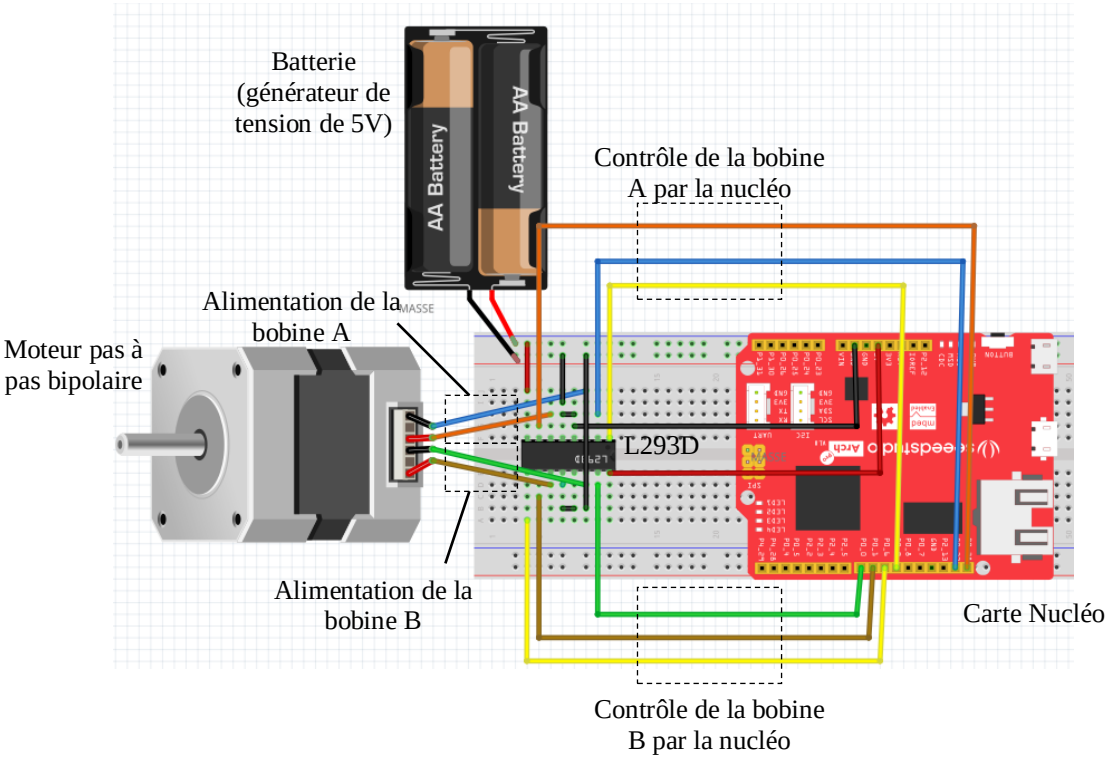


Figure 5 : Schéma du montage permettant de piloter le moteur pas à pas :

Mise en mouvement du moteur

L'axe du moteur fait un pas, soit 1.8°, lorsqu'on alimente une des deux bobines dans un sens. En alimentant successivement chacune des bobines dans un sens puis dans l'autre on fait donc faire quatre pas au moteur dans le sens horaire ou antihoraire. L'ordre d'alimentation des bobines et le sens des courants est important puisqu'il définit le sens de rotation de l'axe mais aussi s'il y a rotation ou non (si on ne les alimente pas dans le bon ordre les rotations peuvent

se compenser et l'axe ne tourne finalement pas). Ainsi, on alimente les bobines dans l'ordre définit ci-dessous à l'aide de la carte Nucléo et du convertisseur de puissance. Pour ce qui est du programme gérant l'alimentation des bobines, on active en continu les broches du convertisseur de puissances commandant chacun des ponts en H (câbles en jaune sur le schéma). Puis, on alimente seulement la broche associée à la bobine et au sens désiré pour alimenter correctement la bobine en plaçant son état à 1 et tous les autres à 0 grâce au fonction *MarcheAV* et *MarcheAR* du programme final (en annexe). Le nombre de pas à réaliser est quant à lui calculer simplement par une règle de trois en fonction de la position finale voulue par la rampe grâce à la fonction *NombreDePas*.

	Bobine 1 sens +	Bobine 1 sens -	Bobine 2 sens +	Bobine 2 sens -
Etape 1	1	0	0	0
Etape 2	0	0	1	0
Etape 3	0	1	0	0
Etape 4	0	0	0	1

	Bobine 1 sens +	Bobine 1 sens -	Bobine 2 sens +	Bobine 2 sens -
Etape 1	0	0	0	1
Etape 2	0	1	0	0
Etape 3	0	0	1	0
Etape 4	1	0	0	0

Figure 6 : Ordre d'alimentation et d'activation des bobines pour faire tourner le moteur dans le sens horaire ou antihoraire :

3) Réalisation du dispositif pour pousser la pièce :

Une fois la pièce détectée, sa valeur identifiée et la bonne boîte sélectionnée en tournant la structure, il faut la pousser dans la rampe transparente. Pour cela, on utilise un servomoteur, qui grâce à une tige en bois en forme d'équerre fixée sur son axe va guider la pièce jusqu'à l'entrée de la rampe. On a choisi un servomoteur standard car il est déjà asservi en angle ce qui permet d'adapter la rotation de la tige à la position de la pièce et la rampe.

Schéma électronique

La mise en place du contrôle angulaire du servomoteur est assez triviale. On connecte simplement les broches du moteur à la carte nucléo. Une broche correspond à la masse, une autre est la tension constante d'alimentation VCC et la dernière est la tension de commande en PWM. Une fois branché, on doit coder l'algorithme qui permet au moteur de répondre au cahier des charges.

Le pilotage du servomoteur se fait par modulation de largeur d'impulsion dans la fonction *PousserPiece* du code en annexe : on envoie un signal rectangulaire dont la largeur varie, ce qui fait changer l'angle d'inclinaison. Deux boucles `for` ordonnent au servomoteur de faire un seul aller-retour (au lieu d'un mouvement de va et vient permanent). Nous avons choisi une durée de pas de 10 ms, soit 10ms*100=1s pour l'aller et 1s pour le retour, ce qui donne une vitesse de poussée convenable (ni trop lente, ni trop rapide ce qui risquerait d'éjecter la pièce).

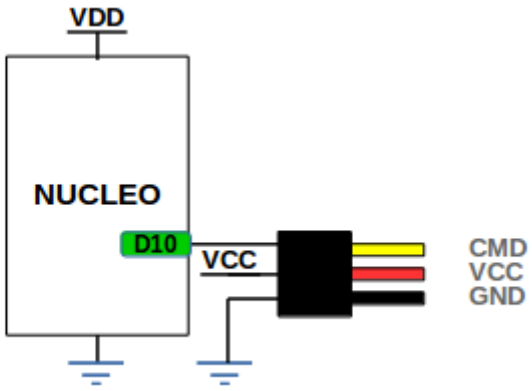
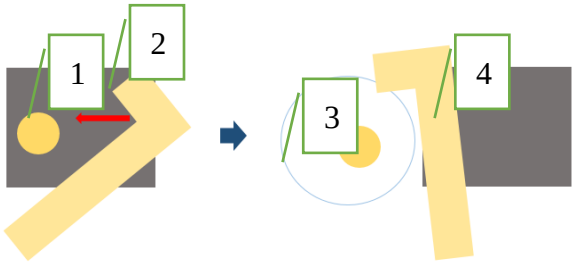


Figure 7 : Schéma de branchement permettant de piloter le servomoteur :

Schéma du dispositif pour pousser la pièce



1 : pièce à trier
2 : balance
3 : tube en plastique transparent
4 : tige pour pousser la pièce

Figure 8 : Schéma de la tige poussant la pièce dans le tube en plastique transparent :

4) Illumination de la machine :

Pour notre projet, on souhaitait porter une attention particulière à l'aspect visuel de la machine. On a donc décidé d'illuminer la rampe de la trieuse en fonction de la valeur de la pièce détectée grâce à un ruban de LED. Il fallait être capable de fournir huit couleurs différentes, une par valeur de pièce, nous avons donc opté pour un ruban de LED trichrome.

Branchement du ruban

Le ruban de LED utilisé contient trois LEDS par ampoule : rouge, vert et bleu. Chacune de ces LED possède son propre circuit de commande composé d'une résistance et d'un transistor MOFSET du type BS170. Le transistor possède trois broches, Grille (G), Drain (D) et la Source (S), il permet d'amplifier le courant fournit aux LEDS. Dans chacun des circuits, la LED est reliée au Drain, la Grille est reliée à une broche PWM de la carte Nucléo pour pouvoir varier son intensité, la source est mise à la masse et une résistance, qu'on choisit à 150 Ohm pour ne pas limiter le courant dans la LED ni la faire griller, est reliée entre la Grille et la masse. Le ruban doit être alimenté par une tension extérieure, il est donc également relié à un générateur de tension lui fournissant 10V.

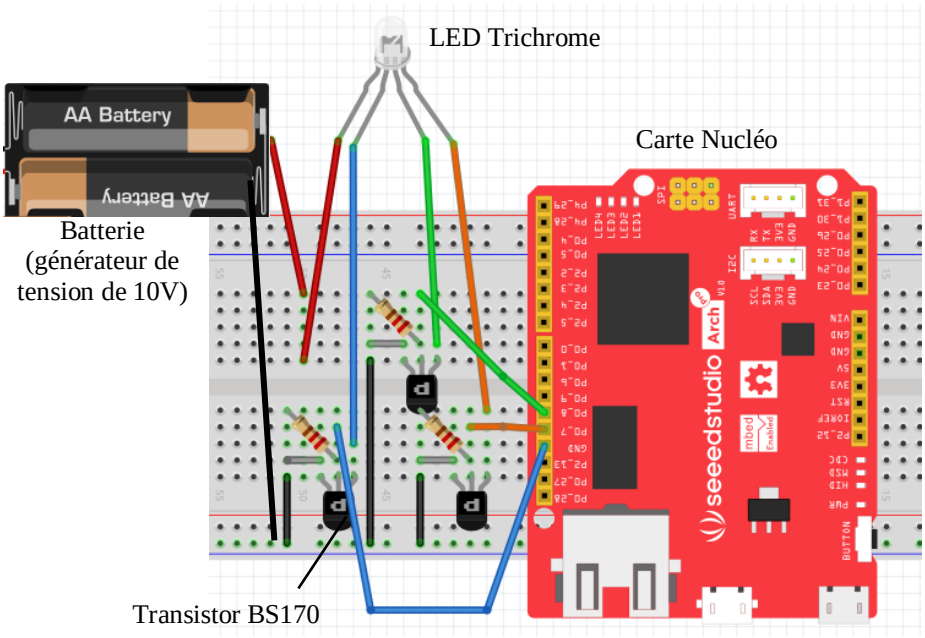


Figure 9 : Schéma du branchement permettant de contrôler le ruban de LEDS :

Pilotage des LEDS en intensité

Pour obtenir nos différentes couleurs on combine les trois LED en variant l'intensité de chacune. Pour cela on les pilote en PWM, c'est à dire qu'on fait varier le rapport cyclique lors de l'activation de la broche qui commande les couleurs. Ainsi, plus le rapport cyclique est faible, moins l'intensité de la LED est élevée, on peut donc mélanger les couleurs pour en former de nouvelles. Nous associons à chaque valeur de pièce une couleur différente grâce aux rapports cycliques adéquats définit ci-dessous grâce à la fonction *Illumination* du code en annexe.

	1ct	2ct	5ct	10ct	20ct	50ct	1€	2€
Rouge	0.5	0.5	0	0.6	0	0.5	0.5	0
Vert	0.1	0.3	0.4	0.3	0	0	0	0.5
Bleu	0	0.1	0.2	0	0.5	0	0.3	0
Couleur	**Orange**	**Blanc**	**Cyan**	**Jaune**	**Bleu**	**Rouge**	**Violet**	**Vert**

Figure 10 : association des couleurs aux pièces et valeurs des rapports cycliques correspondants :

5) Réalisation de la structure :

La plupart de la structure est réalisée par découpe laser de bois, seule la rampe est un tube en plexiglas transparent.

La structure doit avoir une forme fonctionnelle pour supporter tous les éléments qui la compose. On place donc sur le plateau supérieur (2) tous les éléments électroniques qui permettent le contrôle de la trieuse. On fixe le moteur pas à pas directement en bas de la tige centrale pour pouvoir faire tourner la partie supérieure facilement. On ne la fait pas trop grande pour faciliter

la rotation, et creuse pour pouvoir faire passer tous les câbles d'alimentation vers les générateurs à l'intérieur de celle-ci. En effet, pour ne pas encombrer davantage le plateau supérieur, on place les branchements vers les alimentations sous le plateau inférieur avec le moteur pas à pas.

Schéma de la structure

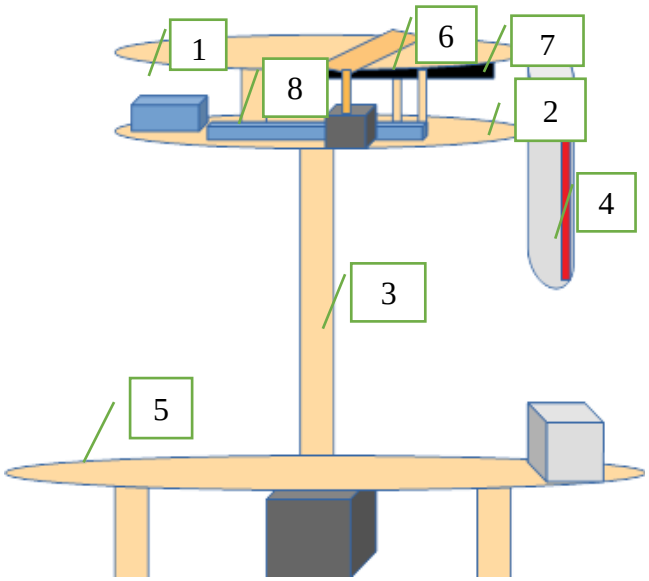


Figure 11 : schéma de la structure de la trieuse de pièces

1 : plateau supérieur supportant la balance, la tige, le tube

2 : plateau supérieur supportant tous les circuits, la carte Nucléo, le servomoteur, le tube

3 : tige centrale qui est solidaire des deux plateaux supérieurs et qui contient les fils d'alimentation

4 : tube en plexiglas permettant de faire tomber la pièce dans sa boîte associée + ruban de LED

5 : plateau inférieur, immobile, sur lequel reposent les boîtes de triage + boîte

6 : servomoteur + tige pour pousser la pièce

7 : balance

8 : circuits + carte nucléo

Design final de la trieuse de pièce

La trieuse de pièce a l'allure espérée à l'origine du projet. Seuls, les câbles reliant la Nucléo à l'ordinateur sortent de la trieuse, la carte n'est en effet pas autonome en énergie. De plus, on peut fermer l'espace entre les deux plateaux supérieurs pour ne plus apercevoir l'ensemble de circuit de commande et rendre la trieuse plus esthétique.

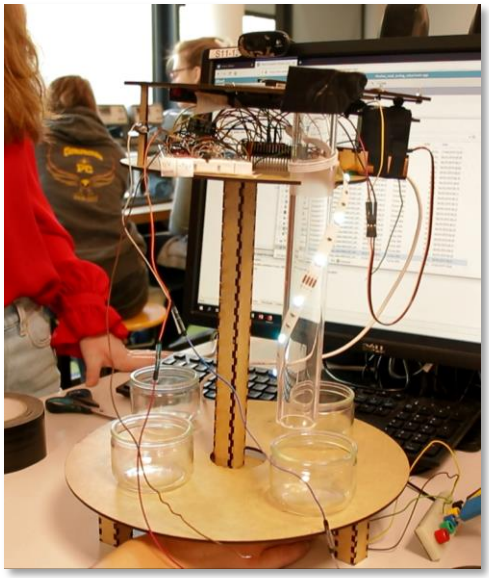


Figure 12 : Photo de la trieuse de pièce à la fin du projet :

III. Système final : La trieuse de pièce

1) Utilisation :

Le système de détection de la pièce ne fonctionnant pas encore à ce stade de la conception, la trieuse ne peut pas identifier la valeur de la pièce mais elle peut néanmoins la trier. Ainsi, pour utiliser la trieuse de pièces il faut suivre les étapes suivantes :

- Poser la pièce à trier sur la balance au niveau du plateau supérieur
- Entrer sa valeur en euro à l'ordinateur dans le code principal (ligne 88 en annexe)
- Le plateau supérieur tourne seul de l'angle adéquat pour positionner la rampe au-dessus de la bonne boite.
- La tige pousse la pièce dans la rampe et celle-ci tombe dans la boite.
- Le plateau supérieur tourne à nouveau, du même angle mais en sens inverse pour se positionner à sa position initiale et permettre le tri d'une autre pièce.

2) Difficultés rencontrées

Dans la détection de la pièce

Qualité du signal et traitement :
Nous avons rencontré des difficultés pour détecter les différentes valeurs de pièces. En effet, il fallait tout d'abord récupérer le signal « brut » issu de la balance. Pour ce faire, il a fallu l'amplifier par un circuit électronique présenté précédemment. Mais même après filtrage, on amplifiait un bruit qui noyait le signal utile. Toute la difficulté a été d'extraire ce signal. Malheureusement les méthodes classiques de traitement de signal (filtrage / amplification) se sont révélées insuffisantes et nous ne sommes pas arrivé à réduire efficacement le bruit, si bien que la détection précise d'une pièce était souvent compromise.

Améliorations :
Une première idée serait d'utiliser une balance type pesées de chimie (assez chère malheureusement), balance dont la tension de sortie est peu bruitée et extrêmement stable. On pourrait éventuellement appliquer quelques traitements numériques (Nucléo) pour calculer une valeur moyenne locale du signal. Mais le seul fait de démarrer avec un signal à peu près propre améliorerait grandement nos résultats.

Une autre idée serait de venir directement prélever le signal en sortie de la balance, et non pas sur la jauge de contrainte. En effet, les balances sont toutes équipées de circuits imprimés qui ont justement pour but de traiter le signal de la jauge de contrainte. En travaillant avec le signal de sortie on retrouverait ainsi toute la précision initiale de la balance.

Amplification :
Même si en théorie, les amplificateurs opérationnels en régime linéaire peuvent amplifier un signal avec n'importe quel gain, on voit en pratique des comportements étranges apparaitre quand on travaille avec des gains trop élevés. Ainsi, le principal problème rencontré en sortie de la nucléo a été l'amplification x100. En théorie un montage amplificateur non inverseur va amplifier l'entrée selon : $Vs = Ve(\frac{R_2}{R_1} + 1)$. La première idée qui était de choisir des résistances

de tel sorte à obtenir un rapport $\frac{R_2}{R_1} \simeq 100$ a révélé que la réelle amplification obtenue (si on en obtenait une) était plutôt de l'ordre de x30. C'est de là qu'est venu l'idée de concaténer deux montages amplificateurs inverseurs avec des gains plus faibles (-10) et qui n'allaient donc pas avoir ce problème.

Dans la motorisation de la structure

Nous avons rencontré des difficultés pour faire tourner correctement la structure sur l'axe du moteur. La présence de jeu au niveau de l'emboitement de la tige sur l'axe provoquait un léger balancement de la tige qui s'amplifiait donc au niveau des plateaux supérieurs. A cause de ce jeu le moteur tournait parfois sur lui-même au lieu de faire tourner la tige.

Amélioration :
On pourrait fixer en plus le moteur à un plateau en bas du moteur pour avoir un support stable qui l'empêcherait de tourner sur lui-même

De plus, le jour de la présentation, le moteur pas à pas n'avait pas le comportement souhaité lors de ses rotations, il faisait un mouvement saccadé et ne tournait donc plus. Nous n'avons pu résoudre ce souci mais il peut être dû à un défaut d'alimentation (celle-ci chutait effectivement fortement lors de la rotation du moteur ce qui n'était pas le cas avant)

Dans la poussée de la pièce

Bien que la rotation de la tige poussant la pièce ait été parfaitement calibrée pour amener la pièce jusqu'à l'entrée de la rampe, celui-ci n'emportait pas systématiquement la pièce. La tige est effectivement surélevée de quelques millimètres par rapport à la pièce et passe donc au-dessus de celle-ci dans certains cas.

Amélioration :
On peut mettre un papier le long de la tige pour combler cet écart. Celui-ci agira comme un balai lorsque la tige tournera et entrainera donc bien la pièce jusqu'à la rampe. Il aurait suffi aussi de trouver une méthode plus optimale pour fixer la balance sous le plateau supérieur (1). En effet, elle se décolle facilement, ce qui crée l'écart entre la tige et la pièce.

Dans la construction de la structure

La structure supportait mal le poids de tous les éléments du projet, principalement placés en hauteur, car la structure tanguait légèrement pendant sa rotation. Malgré que nous ayons mis les câbles du circuit dans la tige pour ne pas encombrer la rotation il restait encore celui de la carte Nucléo.

Amélioration :
On pourrait ajouter une batterie capable d'alimenter la carte et donc de supprimer la connexion à l'ordinateur qui peut gêner à la rotation (il reste des câbles non utilisés dans la tige). On pourrait également réduire la taille de la tige au maximum et augmenter sa largeur pour avoir une surface de contact plus grande avec le plateau supérieur et donc limiter l'effet du mauvais équilibre du poids sur le plateau supérieur lors de la rotation.

ANNEXE :

```
#include "mbed.h"
#define TE          0.01
#define NB_POINT    128
#define SAMPLE_RATE  1/TE

Serial pc(USBTX, USBRX);

//Déclaration des broches de contrôle du moteur pas à pas:
PwmOut bobineA(D11); //bobine A (broche PWM)
PwmOut bobineB(D10); //bobine B (broche PWM)
DigitalOut B_A1(D14);  //Bobine A sens courant 1
DigitalOut B_A2(D15); //Bobine A sens courant 2
DigitalOut B_B1(D9); //Bobine B sens courant 1
DigitalOut B_B2(D8); //Bobine B sens courant 2

//Déclaration des broches de contrôle des LEDS RVB du ruban (broches PWM)
PwmOut Rouge(D5);
PwmOut Bleu (D6);
PwmOut Vert (D3);

//Declaration des broches pour le servomoteur:
PwmOut servo_mot(PC_8); //broche PWM

//Déclaration variables balances:
AnalogIn analog_in(A0);         // A0 on Arduino Header
AnalogOut analog_out(PA_5);     // PA_5 = SCK = D13
Ticker tik;
/* Timer a repetition */
AnalogIn entree_analog(PA_5);
AnalogIn moyenne_amplifiee(A3);
DigitalOut clk_test(PA_6);


//fonction pour illuminer la machine:
void Couleur(double rouge, double vert, double bleu);
void Illumination(double valeur_piece);

//fonction pour la rotation de la plateforme superieure (moteur pas à pas):
void EtatBobineMoteur(int pin1, int pin2, int pin3, int pin4);
void MarcheAV(double attente, int nombre_de_pas);
void MarcheAR(double attente, int nombre_de_pas);

//fonction de calculs intermediaires:
int NombreDePas(int posfin, int posini, int* sens);
int PositionFinale(double valeur_piece);

//fonction pour pousser la piece dans la rampe
void PousserPiece();

//fonction de détection de la valeur de la piece
void convert();
double DeterminerPiece();
```

```
int main()
{
    int PosIni, PosFin, NbPas;
    double ValeurPiece;
    int* Sens;
    int valeur_sens=0;


    Sens=&valeur_sens; //initialisation du pointeur sens

    pc.baud(9600);


    // Activation constante des bobines A et B pour pouvoir les modifier en direct
    bobineA.period(0.02);
    bobineB.period(0.02);
    bobineA.write(1);
    bobineB.write(1);

    //Definition de la période de réglage des intensités en PWM pour ne pas voir cl
ignoter les LEDS.
    Bleu.period_ms(10);
    Rouge.period_ms(10);
    Vert.period_ms(10);

    Couleur(0,0,0); //initialisation du tous les rapports cycliques associés aux LE
DS à 0.

    PosIni=1; //initialisation de la position de la rampe qui correspond à 1ct.

    //On entre la valeur de la piece à detecter si le systeme de detection ne fonct
ionne pas sinon on l'initialise à 0
    ValeurPiece=0.20;

    while(1){

    //Determination de la valeur de la piece avec la balance
    //ValeurPiece=DeterminerPiece();

        if (ValeurPiece!=0){

        //s'il n'y à pas de piece detectée
            if (ValeurPiece==0.01){
                Illumination(ValeurPiece);
                PousserPiece();
                wait(2);
                }
       //si une piece est detectée
            else{
                Illumination(ValeurPiece);      //illumination de la rampe
                PosFin=PositionFinale(ValeurPiece);    //determination de la positi
on finale et du sens de rotation de la rampe
                NbPas=NombreDePas(PosFin,PosIni,Sens);  //calcul du nombre de pas

                //rotation de la structure et descente de la piece
                if (*Sens==-1){
                    MarcheAR(0.07,NbPas); //on tourne en sens trigo
                    wait(1); //temps d'attente apres rotation structure
                    PousserPiece(); //pousser la piece
                    wait(1); //temps d'attente
                }
                else{
```

```
                    MarcheAV(0.07,NbPas);   //on tourne en sens horaire
                    wait(2);
                    PousserPiece();
                    wait(2);
                }

                //retour de la rampe à la position initiale apres descente de la pi
ece
                if (*Sens==-1){
                    MarcheAV(0.03,NbPas);
                    }
                else{
                    MarcheAR(0.03,NbPas);
                    }
            }
        }
        ValeurPiece=0;  //il n'y a plus de piece detectée
    }
}



/*EtatBobineMoteur: permet de faire passer ou non un courant dans les bobines en le
s activant ou non.
Entrées: pin1, pin2, pins3, pin4 : entier valant 0 si on veut desactiver les bobine
s ou 1 si on veut les activer
        elles correspondent successivemement à BobineA sens1, BobineA sens2 et Bobi
neB sens 1, BobineB sens 2
Sortie: aucune*/

void EtatBobineMoteur(int pin1, int pin2, int pin3, int pin4)
{
    B_A1 = pin1;
    B_A2 = pin2;
    B_B1 = pin3;
    B_B2 = pin4;
}



/*MarcheAV: permet de faire tourner le moteur pas à pas à une vitesse souhaitée et
un nombre de pas voulu dans le sens horaire
ENTREES: attente: nombre flottant correspond au temps d'attente en seconde entre de
ux changements d'etat des bobines, permet de controler la vitesse
         nombre_de_pas: entier désigant le nombre de pas effectué par le moteur, 1
pas= 2°
SORTIES: aucune*/

void MarcheAV(double attente, int nombre_de_pas)
{
    for (int i=0; i < nombre_de_pas; i++) {
        EtatBobineMoteur(1,0,0,0);
        wait(attente);
        EtatBobineMoteur(0,0,1,0);
        wait(attente);
        EtatBobineMoteur(0,1,0,0);
        wait(attente);
        EtatBobineMoteur(0,0,0,1);
        wait(attente);
    }
}
```

```
/*MarcheAR: permet de faire tourner le moteur pas à pas à une vitesse souhaitée et
un nombre de pas voulu dans le sens trigo
ENTREES: attente: nombre flottant correspond au temps d'attente en seconde entre de
ux changements d'etat des bobines, permet de controler la vitesse
         nombre_de_pas: entier désigant le nombre de pas effectué par le moteur, 1
pas= 2°
SORTIES: aucune*/

void MarcheAR(double attente, int nombre_de_pas)
{
    for (int i=0; i < nombre_de_pas; i++) {
        EtatBobineMoteur(0,0,0,1);
        wait(attente);
        EtatBobineMoteur(0,1,0,0);
        wait(attente);
        EtatBobineMoteur(0,0,1,0);
        wait(attente);
        EtatBobineMoteur(1,0,0,0);
        wait(attente);
    }
}



/*PositionFinale: associe la position finale de la oiece en fonction de sa valeur
ENTREES: valeur_piece: flottant donnant la valeur de la piece à trier
SORTIE: posfin: position finale de la piece (entier de 1 à 8 correspondant à une de
s 8 boites)*/

int PositionFinale(double valeur_piece){
    int posfin;
    if (valeur_piece==0.01) posfin=1;
    if (valeur_piece==0.02) posfin=2;
    if (valeur_piece==0.05) posfin=3;
    if (valeur_piece==0.10) posfin=4;
    if (valeur_piece==0.20) posfin=8;
    if (valeur_piece==0.50) posfin=6;
    if (valeur_piece==1.00) posfin=7;
    if (valeur_piece==2.00) posfin=5;
    return posfin;
}



/*NombreDePas: calcul le nombre de pas optimal à effecuter pour aller à la position
 finale en partant d'une position initile
ENTREES: posfin: entier contenant la position finale ou doit s'arreter la rampe
         posini: entier contenant la position initiale de la rampe
         marche: pointer vallant 1 si la marche avant est plus courte et -
1 si la marche arrier est plus courte
SORTIE: nbpas: nombre de pas à affectuer pas le moteur*/

int NombreDePas(int posfin, int posini, int* sens){
    int nbpas;
    int AR,AV;
    if (posfin>4){
        AR=8+posini-posfin;
        nbpas=AR*6;
        *sens=-1;
        pc.printf("Posfin>4 : %d\n",*sens);
    }
```

```
    else{
        AV=posfin-posini;
        nbpas=AV*6;
        *sens=1;
        pc.printf("autre : %d\n",*sens);
    }
    return nbpas;
}


/*Illumination: eclaire la rampe en fonctino de la valeur de la piece à trier
ENTREES: valeur_piece: flottant contenant la valeur de la piece
SORTIE: aucune*/

void Illumination(double valeur_piece){
    if (valeur_piece==0.01) Couleur(0.5,0.1,0); //orange
    if (valeur_piece==0.02) Couleur(0.5,0.3,0.1); //blanc
    if (valeur_piece==0.05) Couleur(0,0.4,0.2); //cyan
    if (valeur_piece==0.10) Couleur(0.6,0.3,0); //jaune
    if (valeur_piece==0.20) Couleur(0,0,0.5); //bleu
    if (valeur_piece==0.50) Couleur(0.5,0,0); //rouge
    if (valeur_piece==1.00) Couleur(0.5,0,0.3); //violet
    if (valeur_piece==2.00) Couleur(0,0.5,0); //vert
}



/*Couleur: permet d'associer une couleur au ruban de LEDS en controlant l'intensité
 des trois LEDS RVB
ENTREES: rouge, vert, bleu: nombre flottant correspondant au rapport cyclique de co
ntrole de l'intensité respectif du rouge, vert et bleu
SORTIE: aucune*/

void Couleur(double rouge, double vert, double bleu)
{
    Rouge.write(rouge);
    Bleu.write(bleu);
    Vert.write(vert);
}



/*PousserPiece: fait faire un aller retour au servomoteur pour pousser la piece dan
s la rampe
ENTREES: aucune
SORTIE: aucune*/

void PousserPiece(){
    int i, time = 1500;
    servo_mot.period_ms(20);          // Initialisation période
    servo_mot.pulsewidth_us(1500);    // Initialisation en position

    for(i = 0; i < 100; i++){           //Aller
        time = time+10;
        servo_mot.pulsewidth_us(time);
        wait_ms(10);
    }
    for(i = 0; i < 100; i++){           //retour
        time = time-10;
        servo_mot.pulsewidth_us(time);
        wait_ms(10);
```

```
    }
}


/*convert: Moyenne le signal grace à un filtre moyenneur
ENTREES: aucune
SORTIES: aucune*/

void convert(){
    double somme;
    int i = 0;
    double measT[NB_POINT] = {0};
    clk_test = 1;
    /* FILTRE MOYENNEUR */
    somme = 0;
    for(i = 0; i < NB_POINT-1; i++){
        measT[i] = measT[i+1];
        somme += measT[i];
    }
    measT[NB_POINT-1] = analog_in.read(); // Nouvel echantillon
    somme += measT[NB_POINT-1];
    somme = somme / NB_POINT;             // Calcul de la sortie
    analog_out.write(somme);              // Envoi en sortie
    clk_test = 0;
}


/* DeterminerPiece : détermine la valeur de la pièce posée par calcul de fréquences d'apparition
ENTREES : aucune
SORTIE : valeur de la pièce (en euros)
Double DeterminerPiece(){

    float valeur_euros;

    int N_tot = 0 ;    //Nb total d'apparitions
    int N_2euros = 0;  //Nb d'apparitions de 2 euros
    int N_1euro = 0;   //Nb d'apparitions de 1 euros
    int N_20cents = 0; //Nb d'apparitions de 20 centimes
    int N_0euro = 0;   //Nb d'apparitions de 0 euro

    double freq_2euros = 0; //fréquence d'apparition de 2 euros
    double freq_1euro = 0;    //fréquence d'apparition de 1 euros
    double freq_20cents = 0;   //fréquence d'apparition de 20 centimes
    double freq_0euro = 0;     //fréquence d'apparition de 0 euro

    int compteur = 0;

    /* Association de la fonction convert à un timer */
    tik.attach(&convert, TE);

    pc.baud(9600);  // il faut que cette valeur (=débit de symboles) soit la même
                                                        que celle dans TeraTerm
    int meas_int;
    double tension;

    while(compteur < 5) {
        meas_int = moyenne_amplifiee.read_u16();  // sur 12 bits MSB
        //pc.printf("Tension = %d \n", meas_int);               // Permet d'afficher la valeur
                                                                  de la tension en binaire
                                                                   (inintéressant ici)
```

```
            tension = meas_int / 65536.0 * 3.3;      // en V
            pc.printf("Tension = %lf V \n", tension);                 // Permet d'afficher la
            N_tot+=1;                                                 tension de sortie à chaque fois

            if (tension > 0.594){
                 valeur_euros=2;
                 pc.printf(" *** 2 euros *** \n ");
                 N_2euros = N_2euros + 1;
                 freq_2euros = N_2euros*1.0/N_tot;                          // Permet de calculer
            }                                                        la fréquence d'apparitions de 2 euros

            else if  (tension > 0.587 && tension < 0.592){
                 valeur_euros=1;
                 pc.printf(" *** 20 centimes*** \n ");
                 N_1euro = N_1euro + 1 ;
                 freq_1euro = N_1euro*1.0/N_tot;                            // Permet de calculer
            }                                                          la fréquence d'apparitio ns de 1 euro


            else if  (tension < 0.061 && tension > 0.060){      // Permet de détecter des pièces
 valeur_euros=0.5;
de 20 centimes (mais c'est peu précis)
               pc.printf(" *** 20 centimes *** \n ");
               N_20cents = N_20cents + 1;
               freq_20cents = N_20cents*1.0/N_tot;
          }

          else {
               valeur_euros=0;
               pc.printf(" *** 0 euro *** \n ");
               N_0euro = N_0euro + 1;
               freq_0euro = N_0euro*1.0/N_tot;
               pc.printf("N_0euro= %d \n",N_0euro);               // Permet de calculer la fréquence
          }                                                         d'apparitions de 0 euro

          pc.printf("freq_2euros =%lf \n",freq_2euros);     // permet d'afficher les fréquences
          pc.printf("freq_1euro =%lf \n",freq_1euro);          d'apparition des valeurs
          pc.printf("freq_20cents =%lf \n",freq_20cents);
          pc.printf("freq_0euro =%lf \n",freq_0euro);

          wait(1.0);
          compteur = compteur + 1;
     }

     if (freq_2euros>0.7){       // on choisit d'afficher la valeur de la pièce quand la
          valeur_euros=2;       fréquence d'apparition est > à un seuil (0.7 ici)
     }
     else if (freq_1euro > 0.7){
          valeur_euros=1;
     }
     else if (freq_20cents > 0.7){
          valeur_euros=0.5;
     }
     else if (freq_0euro > 0.7){
          valeur_euros=0;
     }
return valeur_euros ;
}
```