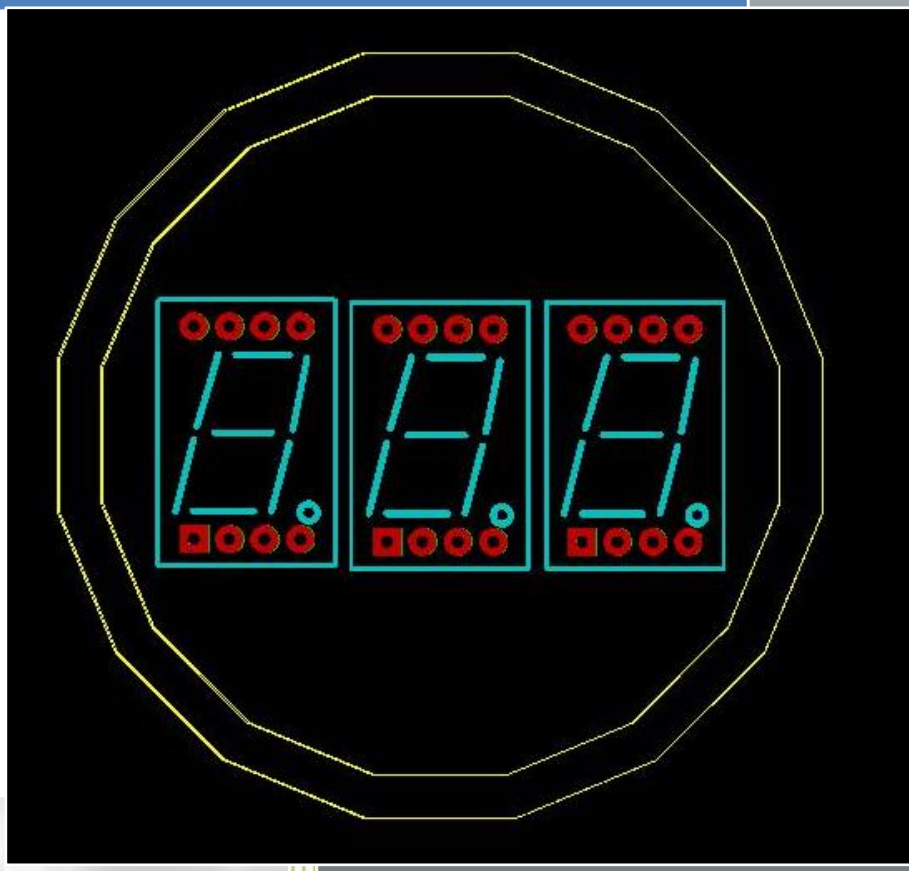
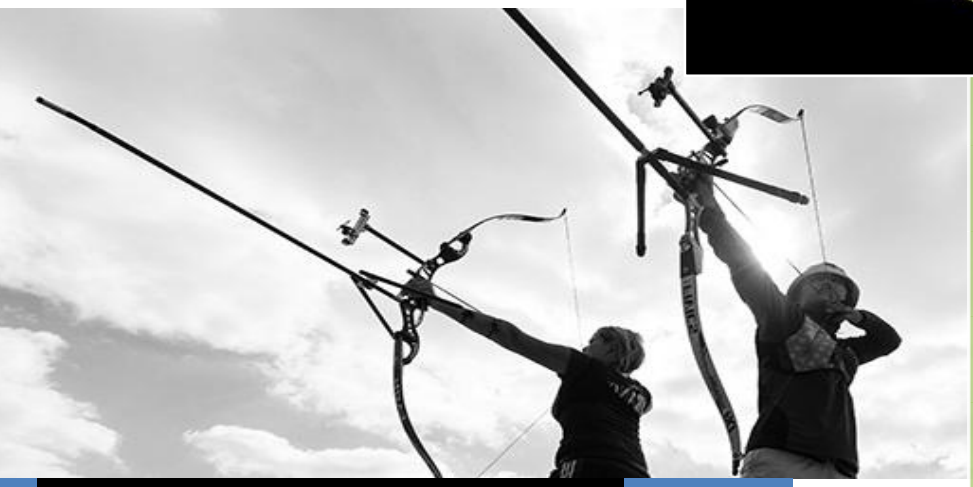
Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Electrique et Informatique Industrielle

E & R

Compteur de flèches





Alexandre Mesland

Groupe : K4A

Année 2013 /2014

Enseignants :
Thierry LEQUEU
Philippe AUGER

Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Electrique et Informatique Industrielle

Compteur de flèches

Alexandre MESLAND
Groupe : K4A
Année 2013/2014

Enseignants :
Thierry LEQUEU
Philippe AUGER

Sommaire

Introduction

Au cours du quatrième semestre, nous devons réaliser un projet d'étude et réalisation, seul ou par binôme. Ayant un projet qui était propre à ma passion qui est le tir à l'arc, je l'ai réalisé seul. Le projet consiste à élaborer un compteur de flèche qui viendrait s'ajouter sur l'arc et afficher le nombre de flèches tirées. Dans ce sport, à haut niveau, il est impératif de connaître la quantité de flèches tirées lors des entraînements. J'ai toujours vu les archers compter leurs flèches tirées sur des petits bouts de papier après chaque tir. Cette idée de compteur me trottait en tête depuis un petit moment déjà et le projet d'étude et réalisation était l'opportunité parfaite pour mettre en œuvre mon projet.

Ce rapport vous montrera les recherches de solutions techniques faites, permettant la réalisation du compteur car ce projet commence de zéro, puis ensuite la création de la carte électronique qui sera enfin pilotée par un programme informatique. Comme vous avez pu le comprendre, le projet se distinguera sous deux parties, une partie électronique et une seconde informatique.

Vous trouverez l'étude théorique qui a été faite afin de lancer le projet et les applications réalisées en expliquant les contraintes rencontrées.

1. Présentation du sujet

Le projet consiste à réaliser un compteur qui pourra s'ajouter sur l'arc et qui, à chaque vibration engendrée par le lâcher d'une flèche, comptera et affichera sur un afficheur le nombre de flèches tirées. Nous sommes dans le cadre d'un projet qui a pour but la mise en application des connaissances acquises lors d'un DUT GEII, mais dans un milieu qui est celui du tir à l'arc, cet objet innovant pourrait être commercialisable s'il arrivait à être léger et compacte.

a) Cahier des charges

- **Origine :** ce choix de projet a été inspiré du besoin de connaitre le nombre de flèches tirées pour un archer lors de son entrainement.

- **But du produit :** compter automatiquement le nombre de flèches tirées par un archer

- **Objectifs de réalisation :**
    - Récupérer l'information qu'une flèche a été tirée
    - Additionner le nombre de fois qu'une flèche a été tirée
    - Afficher le nombre de flèches tirées
    - Avoir un objet indépendant, qui puisse se monter sur un arc

- **Contraintes :**
    - Avoir une alimentation interne
    - Prendre en compte qu'une flèche a été tirée
    - Utilisation d'un microcontrôleur ATmega8535
    - Afficher le nombre de flèches sur un afficheur faible consommation
    - Pouvoir rendre l'objet solidaire de l'arc
    - Résister aux vibrations
    - Etre le moins lourd/encombrant possible

b) Schémas fonctionnel

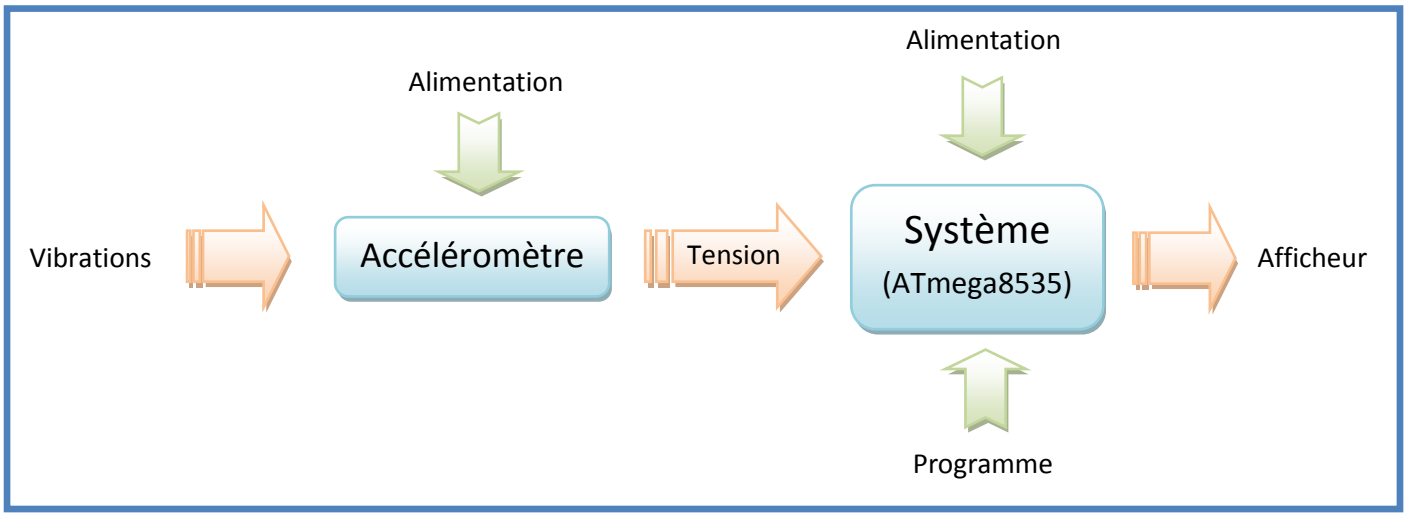


Illustration 1 : Schéma fonctionnel

c) Planning prévisionnel

N° de semaine	37	38	39	40	41	42	43	44	45
Choix du sujet	Prévisionnel							Vacances	
Cahier des charges	Prévisionnel	Prévisionnel						Vacances	
Recherche de solutions	Prévisionnel	Prévisionnel	Prévisionnel					Vacances	
Commande des composants		Prévisionnel	Prévisionnel					Vacances	
Réalisation de la partie récupération d'une vibration			Prévisionnel	Prévisionnel				Vacances	
Programmation du microcontrôleur				Prévisionnel	Prévisionnel	Prévisionnel		Vacances	
Réalisation de la partie affichage						Prévisionnel	Prévisionnel	Vacances	
Confection du boitier							Prévisionnel	Vacances	
Rédaction du rapport et préparation de l'oral							Prévisionnel	Prévisionnel	
Oral								Vacances	Prévisionnel

Prévisionnel | Réel | Vacances

Illustration 2 : Planning prévisionnel

2. Etude des composants

Etant donné que le projet commence de zéro, il a d'abord fallu rechercher des solutions afin de créer un compteur de flèches.

a) Les détecteurs de vibrations

Pour commencer, je me suis interrogé sur quelle information m'appuyer pour considérer qu'une flèche a été tirée. J'ai choisi la vibration émise par l'arc lors du tir d'une flèche, car l'intensité des vibrations produites lors du tir sont difficilement atteignables sans le tir d'une flèche, ce qui fait que cette vibration est pour moi la source d'information la plus sûre pour compter. Cependant, n'ayant jamais utilisé de composant travaillant sur les vibrations, je n'ai pas pu réussir à trouver immédiatement le composant qu'il me fallait.

1) Le détecteur de choc 801S

Pour débuter, j'ai recherché un composant qui réagissait par rapport aux vibrations, après une recherche sur internet, il fut difficile de choisir étant donné mon manque d'expérience dans ce domaine. Je me suis lancé avec le détecteur de choc 801S acheté sur le site Selectronic, car en lisant la documentation constructeur (disponible en Annexe 1) il semblait répondre à mes attentes : il détecte les chocs, il peut être utilisé pour réaliser une alarme, il est assez petit, plutôt résistant, il ne coûte que 3,60€ et le montage d'une application est donné dans la documentation constructeur. Néanmoins, il y a peu d'informations données sur ce composant autres que les sites qui le vendent, donc il était difficile de savoir entièrement à quoi s'attendre.

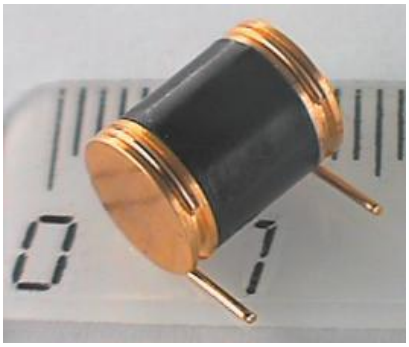


Illustration 3 : détecteur de choc 801S

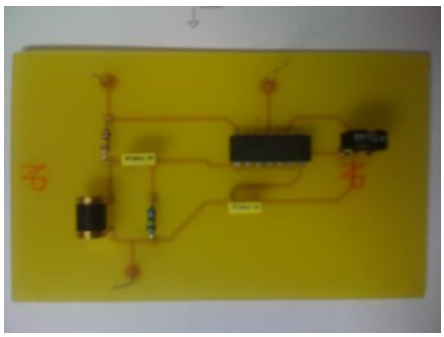


Illustration 4 : montage avec le 801S

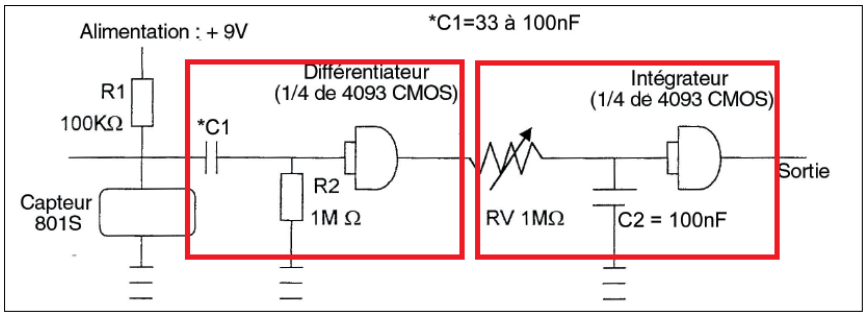


Illustration 5 : Schéma du montage avec le 801S

Une fois le composant reçu, j'ai réalisé une carte électronique du montage fourni avec la documentation constructeur. Il est composé d'une partie différentiateur (un filtre CR) qui fait office de détecteur de front montant et d'une partie intégrateur (filtre RC) avec la sortie variant d'un état haut à un état bas suivant le seuil réglé grâce à la résistance variable.

Après avoir testé cette carte j'ai pu comprendre que le 801S est en fait un contact ouvert qui se ferme en présence de vibration. Ici le capteur réagit dès qu'on le touche, il est extrêmement sensible et nous pose problème. Il ne nous permet pas de filtrer l'intensité de vibration voulue, la sortie se met en état haut alors que nous avons une faible vibration, ce qui veut dire qu'on ne sera pas capable de différentier une grosse vibration qui correspondra au tir d'une flèche. Le capteur se met en isolant juste pour un petit changement de position et ce n'est pas la résistance variable qui va changer quelque chose. Ce composant ne pourra pas être utilisé dans le cas présent car si on le met sur l'arc, dès qu'on saisit l'arc, les vibrations provoquées feront que le compteur sera en train de compter alors qu'aucune flèche n'aura été tirée.

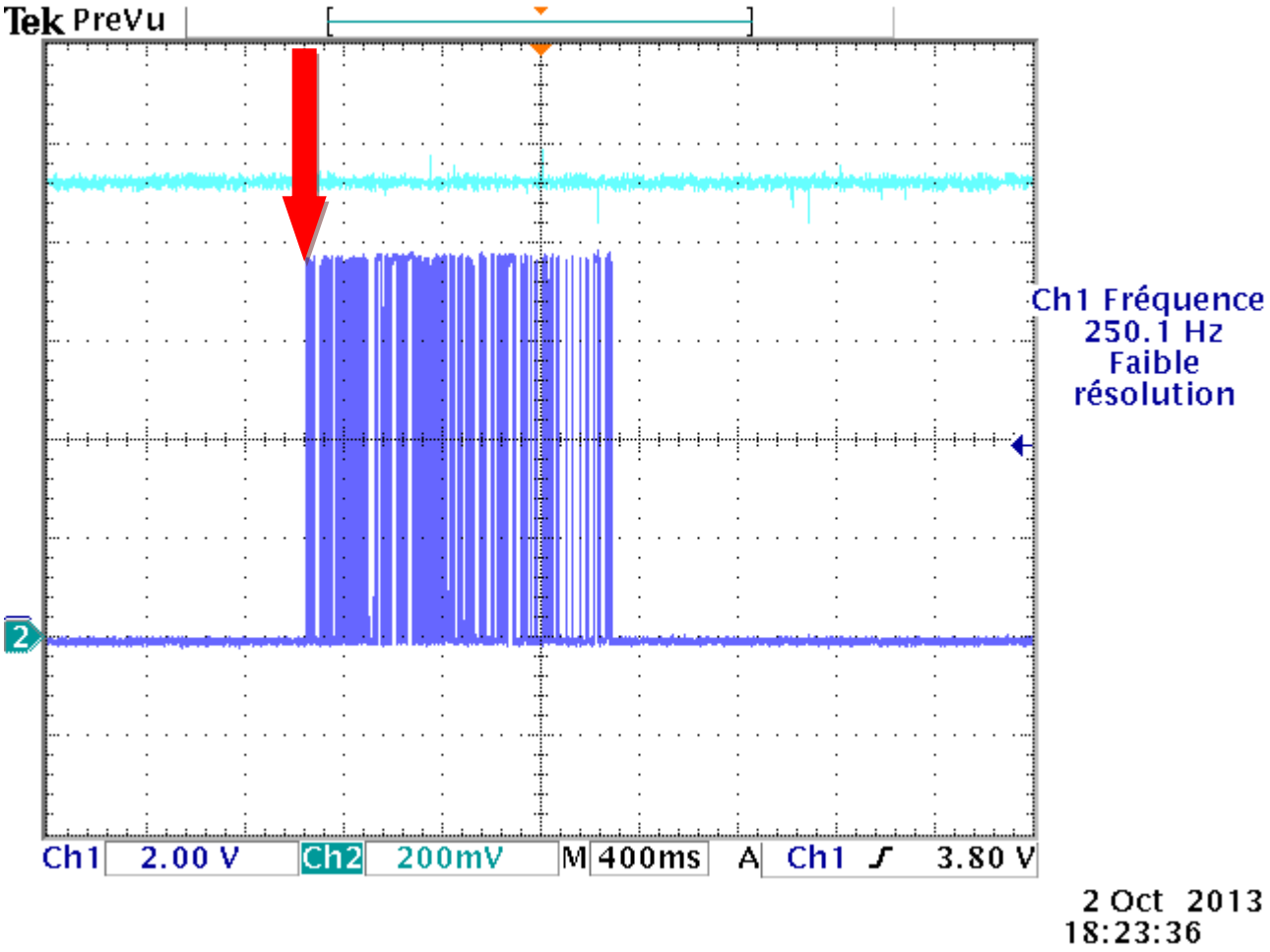


Illustration 6 : Relevé de la sortie du montage du 801S en présence d'une vibration

Pour vous témoigner de la sensibilité de ce montage, sur ce relevé d'oscilloscope[1], la flèche rouge simule le moment ou j'ai donné un petit coup sur la table où est disposé le montage.

[1] Appareil permettant de visualiser un signal électrique

2) L'accéléromètre ACCM2G2

Suite à l'échec avec le détecteur de choc 801S, j'ai commandé un accéléromètre ACCM2G2 (SAURET, 2005) sur le site Gotronic. Ce composant est basé sur un accéléromètre LIS244ALH de ST MicroElectronics qui mesure l'accélération sur 2 axes. Cependant les sorties de ce composant sont de très faible valeur, donc des amplificateurs opérationnels ont été ajoutés autour de celui-ci afin de récupérer des tensions assez élevées pour qu'on puisse les exploiter (Document constructeur disponible en Annexe 2).

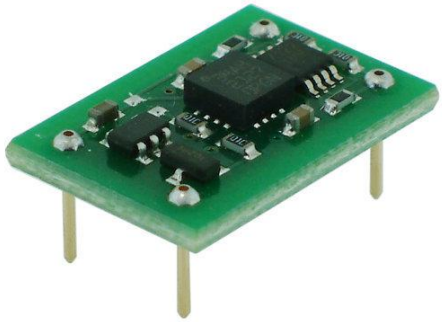


Illustration 7 : Accéléromètre ACCM2G2

Ce composant est composé de quatre bornes :
- une borne d'alimentation Vcc (3,5V < Vcc< 15V)
- une masse GND
- une borne de sortie X donnant une tension proportionnelle à l'accélération en X
- une borne de sortie Y donnant une tension proportionnelle à l'accélération en Y

Les tensions de sortie X et Y seront égales à $\frac{\mathrm{Vcc}}{2} + 660\ \mathrm{mV/g}$

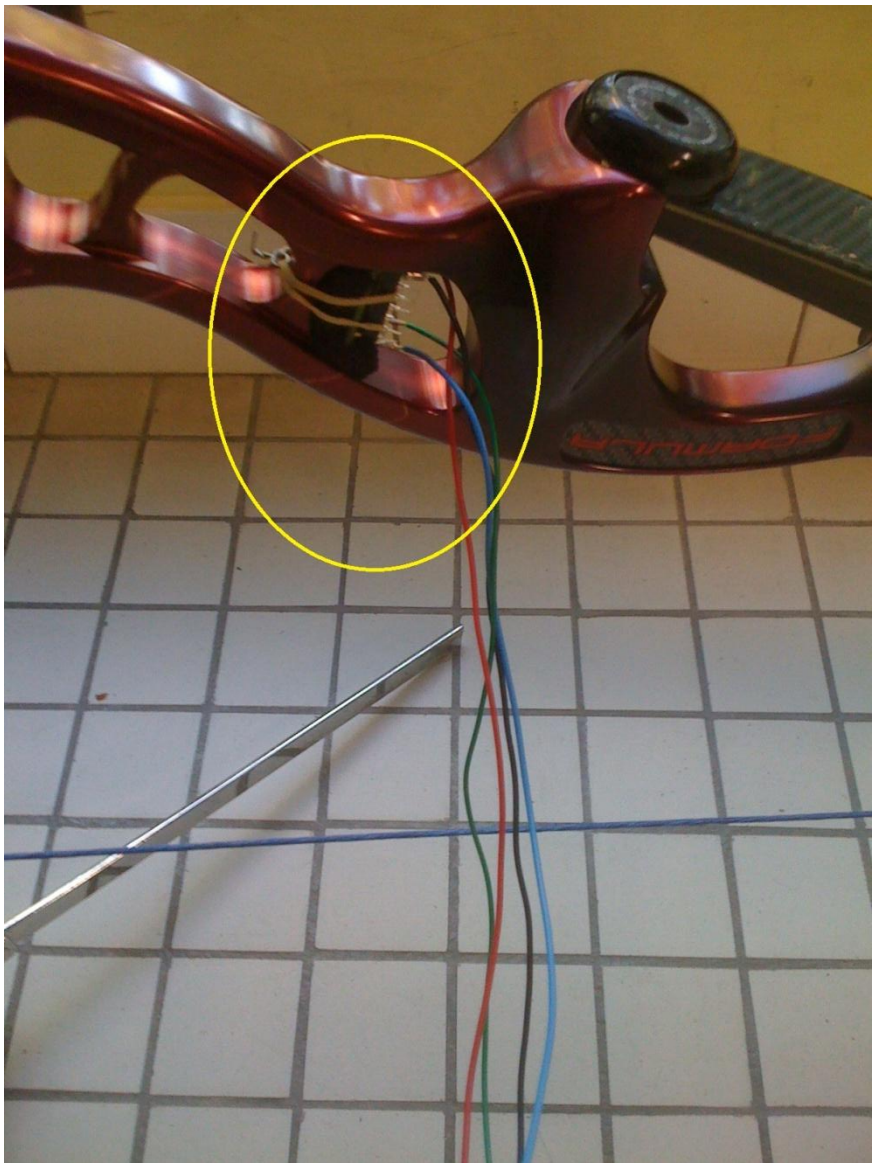


Illustration 8 : Disposition de l'accéléromètre sur l'arc

A savoir : pour ce composant les deux sorties donnent X et Y avec une sensibilité de ±2 g. Si l'accéléromètre subit une accélération de plus de 2g, il ne pourra pas donner une tension supérieure à celle qu'il donne déjà pour 2 g (pour info 1 g = 9,80665 $m.s^{-2}$).

Pour voir si cet accéléromètre est bien celui qui me faut pour mon projet, j'ai relié les quatre bornes de l'accéléromètre à de longs fils (Vcc et GND à une alimentation stabilisée pour alimenter l'accéléromètre et les deux sorties X et Y de l'accéléromètre à un oscilloscope pour visualiser ces les tensions générées lors de vibrations).

Sur l'illustration 8, l'endroit ou a été fixé l'accéléromètre a été entouré en jaune.

J'ai ensuite tenté de simuler les vibrations que procure le tir d'une flèche, il s'est avéré que les sorties du composant saturées à 3 V en effectuant de petites vibrations sur l'arc (voie en bleu sur l'oscilloscope de l'illustration 9). Ce fut problématique, car on est encore loin de l'intensité que va procurer le tir réel d'une flèche. Avec cette sensibilité là, ça veux dire que lorsque que l'on va poser l'arc par terre, la vibration engendrée sur l'arc sera assez forte pour que les sorties de l'accéléromètre saturent. Autrement dit, lorsque l'on pose l'arc par terre ou que l'on tire une flèche, la sortie de l'accéléromètre sera saturée à 3 V. Le problème ici sera donc qu'au point de vue informatique on ne pourra pas faire la différence entre ces deux évènements.

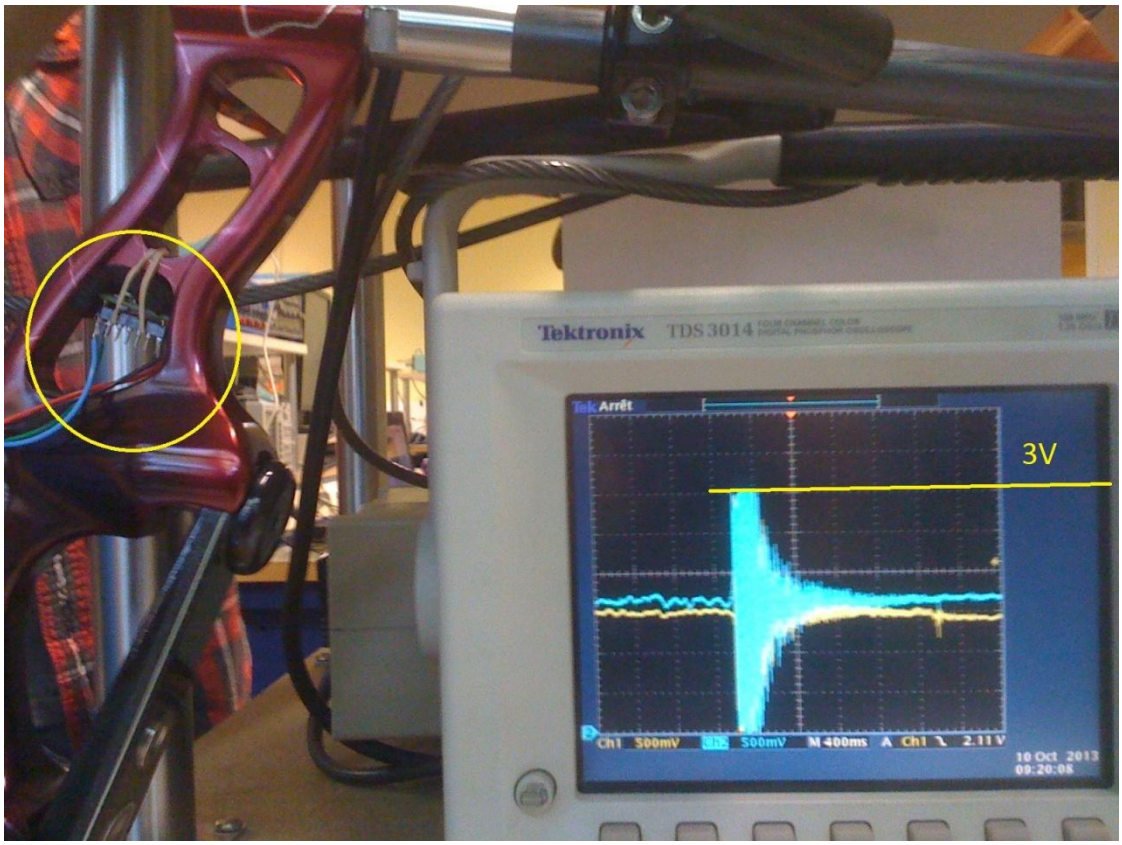


Illustration 9 : Disposition de l'accéléromètre sur l'arc + signaux de sortie de l'accéléromètre

La sensibilité de cet accéléromètre est de ±2 g, c'est-à-dire que le tir d'une flèche provoque des vibrations supérieures à 2 g sur l'arc lors du tir d'une flèche. Déduction, il faut trouver un accéléromètre avec une plus grande sensibilité, afin que l'on puisse repérer de plus grandes accélérations, notamment celle de l'intensité lors du tir d'une flèche.

3) L'accéléromètre ACCM6G

Etant satisfait du principe de fonctionnement du dernier accéléromètre, on a cherché le même mais avec une sensibilité plus élevée. Le fabricant propose deux modèles pour cet accéléromètre : le ±2 g et le ±6 g. Le 6g est une sensibilité trois fois plus grande que le 2g, le choix du 6g parait donc approprié à mon application. Après avoir commandé et reçu, j'ai procédé aux mêmes tests qu'avec le précédent.

Les tests se sont avérés meilleurs, seulement le composant sature encore pour une intensité de vibration qui n'atteint pas l'intensité qui sera obtenue lors du tir d'une flèche. Ce qui veut dire que, contrairement à ce que je pensais, lors du tir d'une flèche, l'arc a une accélération de plus de 6g si on en suit les données du constructeur de l'accéléromètre (attention, je ne confirme pas que l'arc reçoit réellement 6g lors du tir, les données constructeurs de sont pas toujours en phase avec la réalité, des mesures physiques n'ont pas été réalisées sur le tir d'une flèche).

Toujours le même problème, ça veut dire que le composant ne pourra pas faire la comparaison entre une accélération de 6g (selon l'accéléromètre) et l'accélération émise lors du tir d'une flèche, supérieur à 6g (selon l'accéléromètre). Cet accéléromètre nous permet donc de recueillir d'importantes vibrations, mais son utilisation impliquerait la possibilité que d'autres chocs ou mouvements de l'arc soient assez forts pour qu'ils soient comptés comme une flèche tirée. Etant donné le temps passé sur le choix d'un détecteur de vibration et le nombre de séances restantes, je vais continuer avec ce composant, car si l'arc est utilisé sans mouvements brusques entre les tirs, le composant correspondrait aux attentes.

Voici les deux signaux de sorties obtenus suite à la simulation des vibrations engendrés par le tir d'une flèche :

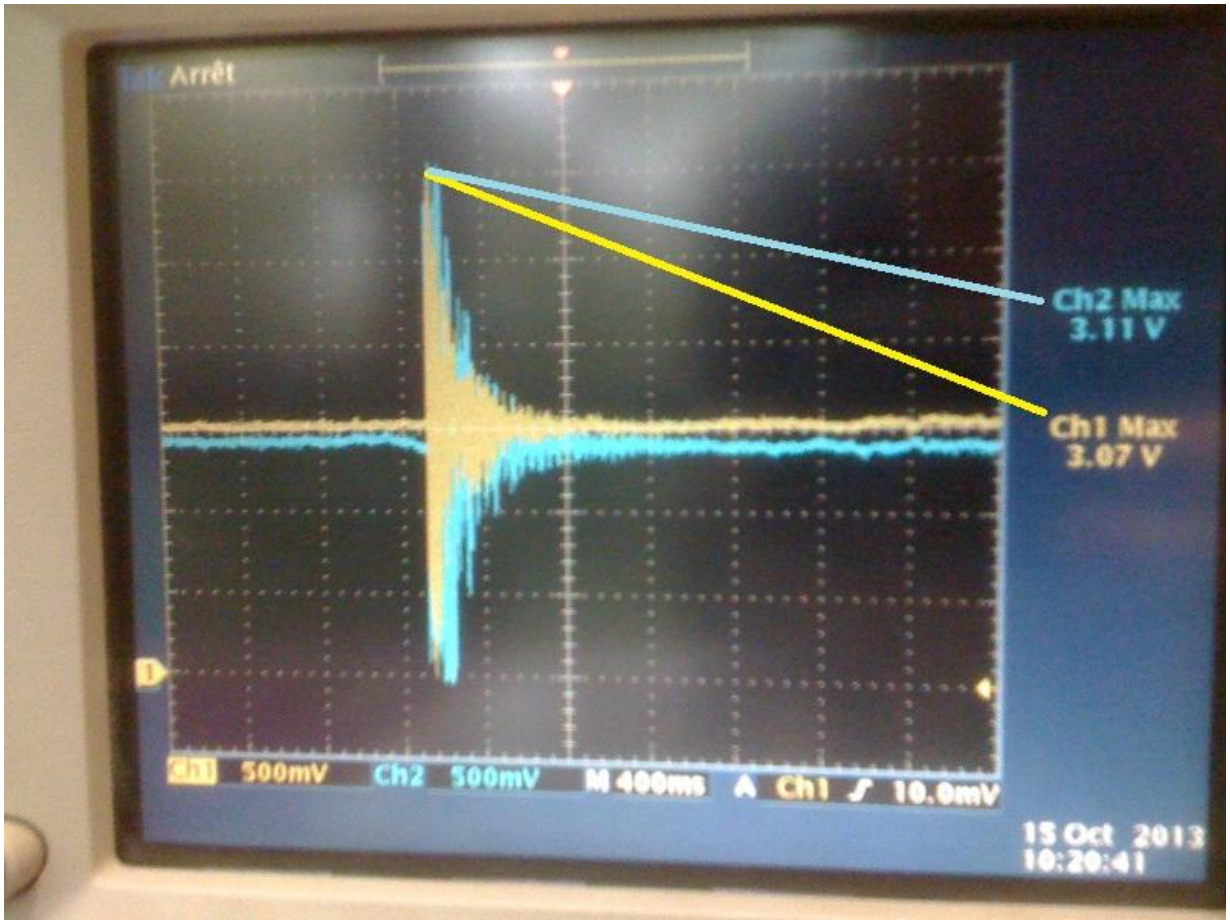


Illustration 10 : Signaux de sortie de l'accéléromètre

b) Le microcontrôleur

Sur la carte finale, le microcontrôleur aura un rôle très important, qui sera de récupérer l'information qu'une flèche aura été tirée, par le biais d'une tension générée par l'accéléromètre, de la traiter à l'aide d'un programme informatique et d'ensuite afficher le nombre de flèche tirées sur des afficheurs.

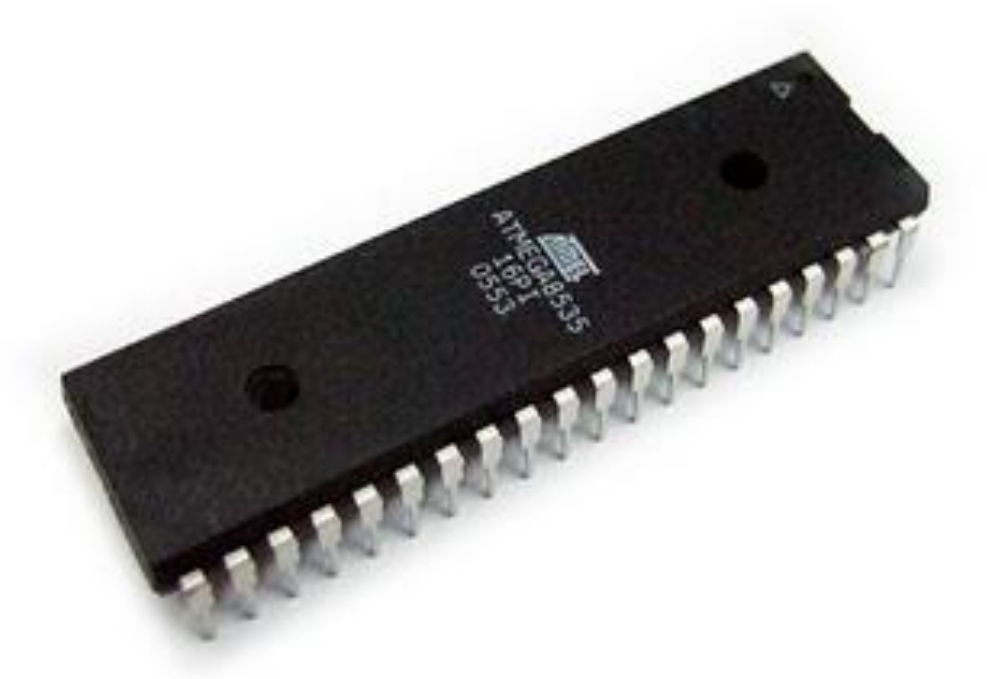


Illustration 11 : ATmega8535

Mon choix s'est porté sur le microcontrôleur ATmega8535 d'Atemel[2]car il possède une mémoire intégrée, permettant d'intégrer directement le programme dans le microcontrôleur, ce qui lui permet d'être autonome une fois le programme implanté. Il est alimenté en 5V, c'est une faible tension qui nous laisse envisager la possibilité de l'alimenter par de simples piles. Un critère, qui m'a permis de trancher sur son choix, c'est qu'il dispose de quatre ports (PORT A, PORT B, PORT D et PORT D) de 8 bits chacun, numérotés de 0 à 7, programmables comme entrée ou comme sortie. Au final l'ATmega8535 comporte 40 broches[3].

[2] Nom du constructeur de l'ATmega8535
[3] Bornes du composant

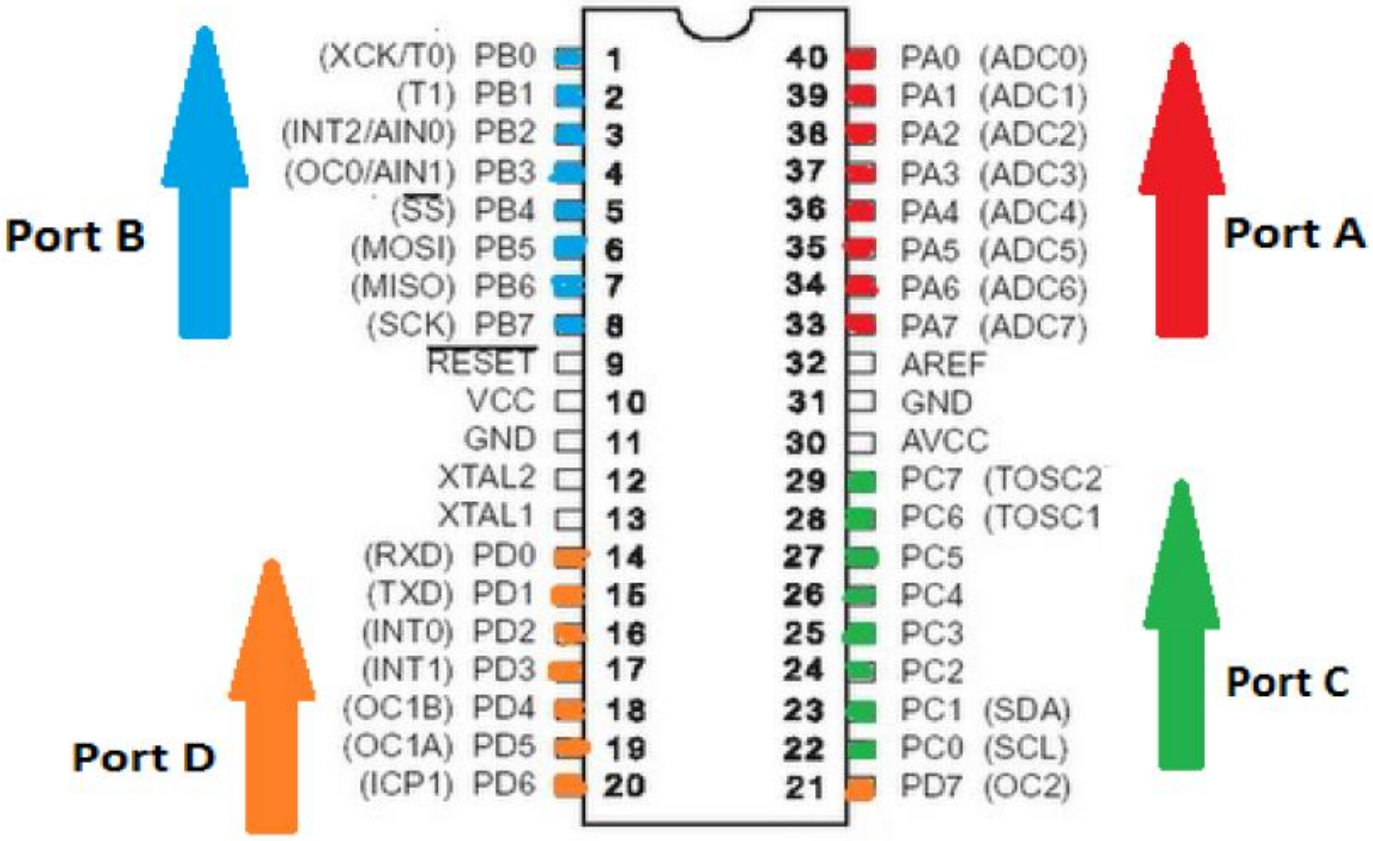


Illustration 12 : ATmega8535 avec ses bornes de détaillées

Le **Port A** (de PA0 à PA7) dispose de 8 entrées (PA0 à PA7) qui peuvent peut être programmées comme Convertisseur Analogique Numérique(CAN). Elles pourront nous être utiles pour recueillir la tension de sortie de l'accéléromètre et également comme sorties pour afficher sur les afficheurs.

AREF est l'entrée de référence analogue pour le CAN.

GND est la masse de l'alimentation.

AVCC est la broche d'alimentation pour le CAN.

Les **Port B**, **C** et **D** serviront de sortie pour afficher sur les afficheurs.

RESET permet de la réinitialisation du microcontrôleur.

VCC est la broche d'alimentation du microcontrôleur (entre 4.5V et 5.5V)

XTAL1 et **XTAL2** sont reliés entre un quartz cadencé à 16MHz pour remplacer l'horloge interne du microcontrôleur pour une meilleure précision de chronométrage.

c) Les afficheurs

L'afficheur recherché, est un afficheur qui se veut petit, mais surtout qui consomme le moins possible dans le but d'une alimentation par pile. La meilleure solution pour des afficheurs qui ne consomment pas, ce sont les afficheurs LCD. Cependant, le nombre amené à être afficher, devra comporter trois chiffres, au pire quatre, car en général lors d'un entraînement, l'archer va tirer plus de 100 flèches, mais il est très peu probable qu'il en tire 1000. Je n'ai pas réussi à trouver d'afficheur LCD 3 digits (=3 chiffres), en 4 digits j'en ai trouvé quelques uns, le problème c'est qu'ils étaient vraiment trop encombrants, de plus ils étaient très difficiles à mettre en application et ils avaient une documentation technique peu suffisante, pour savoir comment s'y prendre.

Mon choix c'est donc porté sur des afficheurs 7 segments[4] à LED[5], au détriment de la consommation. J'ai donc cherché l'afficheur le plus petit possible, avec la plus faible consommation bien que sur ce plan je n'ai pas eu trop le choix. J'ai choisi le FN1-0312B2300 à LED bleu de chez Farnell[6], car il est assez petit (12 .7mm de hauteur pour 7.6mm de largeur) et consomme une vingtaine de milliampères comme la plupart des afficheurs à LED, même si sur le site de Farnell il est précisé qu'elles ne consomment que dix milliampères. Autre information fausse sur le site du vendeur, il y était précisé que ces afficheurs étaient à cathode commune, alors que ce n'était d'ailleurs pas référencé dans la documentation constructeur, en réalité elles étaient à anodes communes.

Illustration 13 : Afficheur FN1-0312B2300

Cet afficheur, est composé de 10 broches, l'afficheur étant à anode commune, il faut alimenter les broches 1 et 6 et puis pour allumer le segment voulu il suffira de mettre à 0 la broche du segment en question et pour éteindre le segment. A noter également que la documentation du constructeur nous dit que la luminosité maximale du segment est atteinte à 25 mA et que la tension préconisée aux bornes de la LED est de 3,2V. Dans mon cas je considère que la luminosité de la LED est assez suffisante pour 20 mA. Pour avoir un courant de 20 mA dans la LED, je vais ajouter une résistance en série avec chaque LED, sachant que l'on veut 3,2V sur la LED, il y aura 1,8V sur la résistance (5 - 3,2 = 1,8). Sachant que l'on veut 20 mA dans la LED il faut dimensionner la résistance en conséquence avec la loi d'ohm. On obtient une résistance de 90 Ω (1,8 / 0,02), en pratique on choisira une résistance de 100 Ω car elle est disponible au magasin de l'IUT.

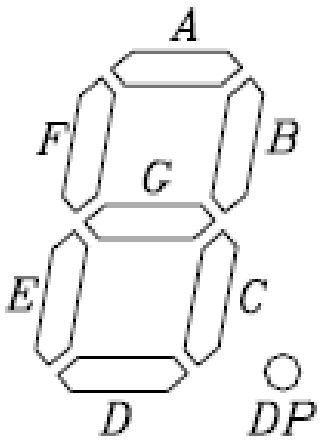


Illustration 15 : Correspondance de chaque segment

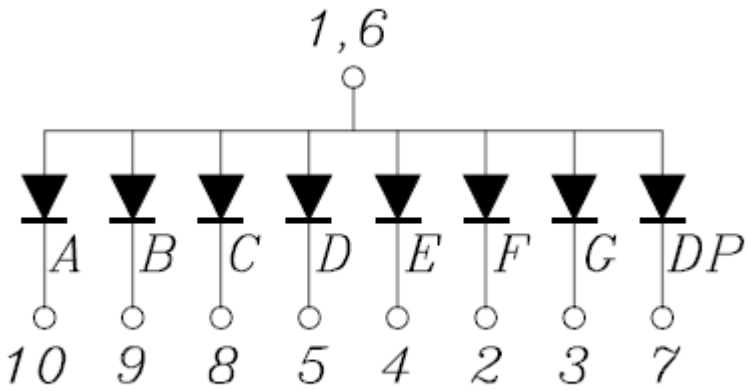


Illustration 14 : Schéma électrique de l'afficheur

[4] Un segment est un petit trait permettant de faire une partie d'un chiffre sur l'afficheur, il faut 7 segments pour faire tous les chiffres possibles

[5] Diode Electro Luminescente

[6] Nom du vendeur

d) Le boitier

Le but est de faire un boitier qui soit solidaire de l'arc (une fois sur l'arc il ne bouge plus), qui résiste aux vibrations dans le temps, qu'il soit le plus léger possible (l'idéal serait que l'archer ne remarque pas au niveau du poids l'ajout du compteur, pour ne pas le gêner dans sa pratique) et si il peut être esthétique cela pourrait être un plus.

Illustration 16 : photos du boitier

Pour réaliser ce boitier, j'ai pris une partie d'une lampe de torche, à laquelle je visse une pièce en aluminium usinée sur mesure dans le but de fermer l'ouverture et à l'intérieur de laquelle je visse un adaptateur de pas de vis permettant de le visser sur un arc. L'objet obtenu, est majoritairement composé d'aluminium, ce qui nous assure une solidité largement suffisante, ainsi qu'un poids raisonnable.

3. Elaboration de la carte électronique

a) Le régulateur +5V, 0,5A

Comme dans un premier temps, nous cherchons à faire une carte prototype cherchant juste à établir le fonctionnement du compteur, on veut établir un système de régulation à l'entrée de l'ATmega8535 pour être sûr qu'il y ait toujours 5V fixe à son entrée quelque soit la tension que l'on apporte à l'entrée du régulateur (à condition qu'elle soit quand même inférieure à 60V).

Le composant utilisé est un régulateur de type DC/DC de référence LM2574-5. Il permet d'obtenir à partir d'une tension d'entrée inférieure à 60V, une tension de +5V et un courant de 0,5A d'une manière constante quelque soit la consommation de la carte car on sait que les principalaux composants consommant du courant sont les afficheurs, à raison de 20 mA par segment d'allumé, sachant qu'au maximum il ne peut y avoir que 3x7 (=21) segments d'allumé à la fois et 21x0,02 = 0,42 A, on reste en dessous des 0,5 A du LM2574-5.

Une LED sera présente en sortie du +5V afin de témoigner qu'il y ait bien une tension présente.

La tension VCC et une tension de 5 V toujours mais encore mieux filtrée, elle est sensée être utilisée pour la broche AREF de l'ATmega8535, comme référence analogique pour le CAN.

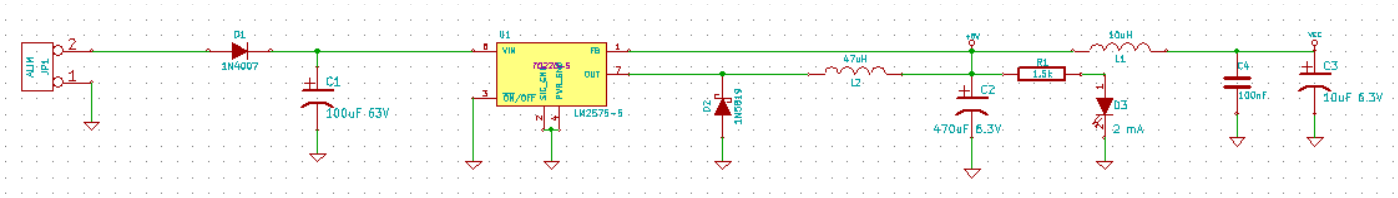


Illustration 17 : Schéma électrique de la partie régulation

b) ATmega8535 avec les afficheurs

Le connecteur P3 correspond à l'accéléromètre, les deux sorties X et Y de celui-ci iront sur les entrées analogiques qui se trouvent sur le port A.

Les autres broches du port A seront configurées en sortie afin de contrôler le premier afficheur, seulement comme deux broches du port A ont été prises comme entrée, il ne reste pas assez de broches sur ce port pour contrôler entièrement le premier afficheur, donc on ira utiliser des broches du port B. A savoir que les deux bornes (1) et (6) d'un afficheur son reliées au +5V et qu'ensuite on a 7 broches qui pourront autoriser l'allumage d'un segment en mettant le bit de sortie à 0 (ce qui laisse passer le courant dans la diode et l'allume) il reste une dernière broche dans l'afficheur, la broche (7) qui correspond au point, mais dans notre cas nous n'avons pas besoin du point donc on relie cette borne à l'alimentation, comme il n'y a aucune différence de potentiel aux bornes de la LED allumant le point, il ne s'allumera pas. Ce qui fait au total 7 broches par afficheur que l'on à besoin de piloter. On relira le 2 ème afficheur à 7 broches du port C et le 3 ème afficheur à 7 broches du port D. Il ne faut pas oublier que chaque sortie de l'ATmega8535 servant à allumer un segment doit comporter une résistance de 100Ω afin qu'il y ait assez de courant pour allumer le segment.

Le connecteur P2 servira à connecter une autre carte qui elle, sera reliée au PC et servira à transférer le programme de l'ordinateur à l'ATmega8535.

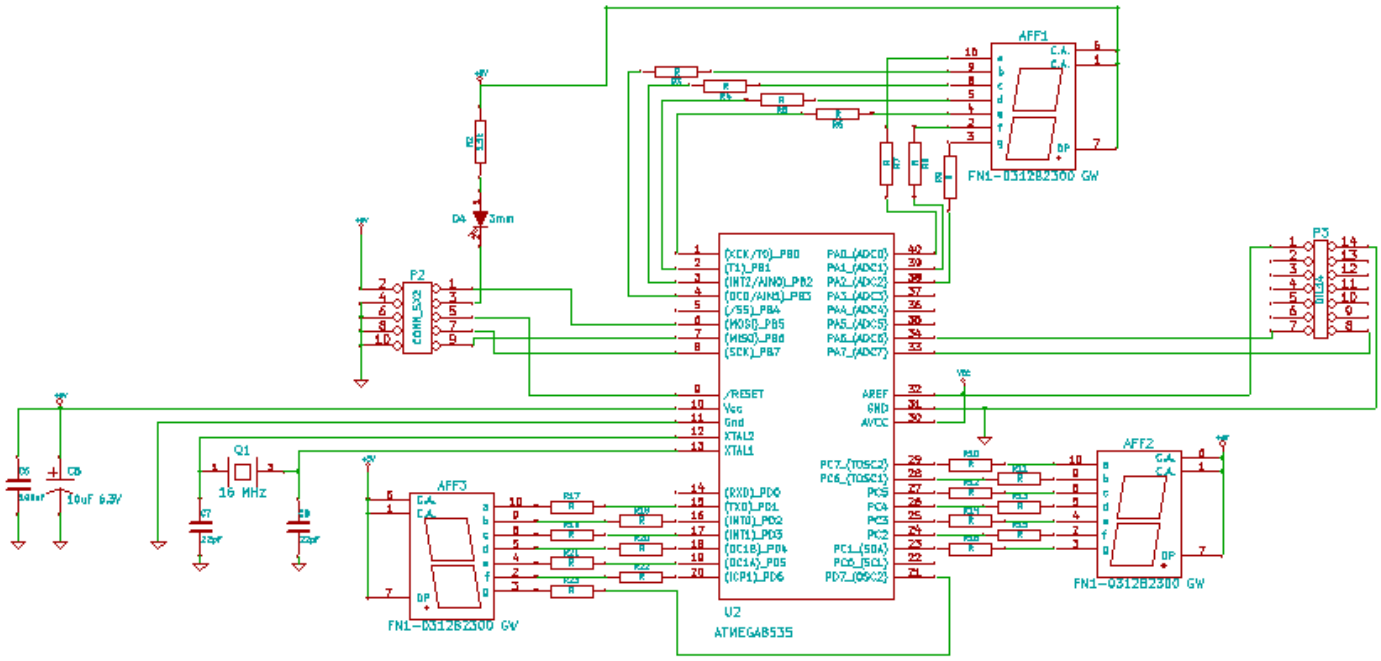


Illustration 18 : Schéma électrique du câblage de l'ATmega8535

c) Liste des composants

Pour la réalisation de la carte électronique, j'ai choisi le logiciel Kicad pour faire le typon. J'ai donc d'abord effectué les schémas électriques vus précédemment, puis j'ai listé mes composants.

N°	Référence	Boitier	Quantité	Désignation	Fournisseur
1	AFF3,AFF2,AFF1	HDSP-78xx	3	FN1-0312B2300 GW	Farnell
2	R2,R1	R4	2	1.5k	IUT
3	R8,R7,R3,R6,R9,R5,R4,R15,R14,R13,R12,R11,R10,R17,R18,R19,R20,R21 R22,R16,R23	R4	21	R	IUT
4	D3	LED-5MM	1	2 mA	IUT
5	D4	LED-3MM	1	3mm	IUT
6	P2	he10-10d	1	CONN_5X2	IUT
7	Q1	HC-18UV	1	16 MHz	IUT
8	U2	DIP-40__600_ELL	1	ATMEGA8535	IUT
9	P3	DIP-14__300_ELL	1	DIL14	IUT
10	D2	D3	1	1N5819	IUT
11	D1	D3	1	1N4007	IUT
12	C6,C4	C2	2	100nF	IUT
13	L2	C2	1	47uH	IUT
14	C8,C7	C1V8	2	22pF	IUT
15	C5,C3	C1V5	2	10uF 6.3V	IUT
16	C2	C1V5	1	470uF 6.3V	IUT
17	C1	C1.5V8V	1	100uF 63V	IUT
18	L1	C1	1	10uH	IUT
19	JP1	bornier2	1	ALIM	IUT
20	U1	DIP-8__300_ELL	1	LM2575-5	IUT

Illustration 19 : liste des composants

d) Routage de la carte

Sur kicad, on commence par faire le schéma électrique, ensuite on génère la Netliste qui permet de créer les composants avec les connexions qu'il y a entre eux. Ensuite, on liste les composants en leur attribuant des empreintes à chacun. Et enfin on lit la Netliste, on place les composants sur une carte puis on trace les pistes afin d'effectuer les connexions entre les composants.

Le but étant de faire une carte disposant les composants de manière stratégique (par bloc) de façon à faire des pistes les moins longues possibles pour éviter les problèmes de CEM[7].

Voici la carte finale obtenue :

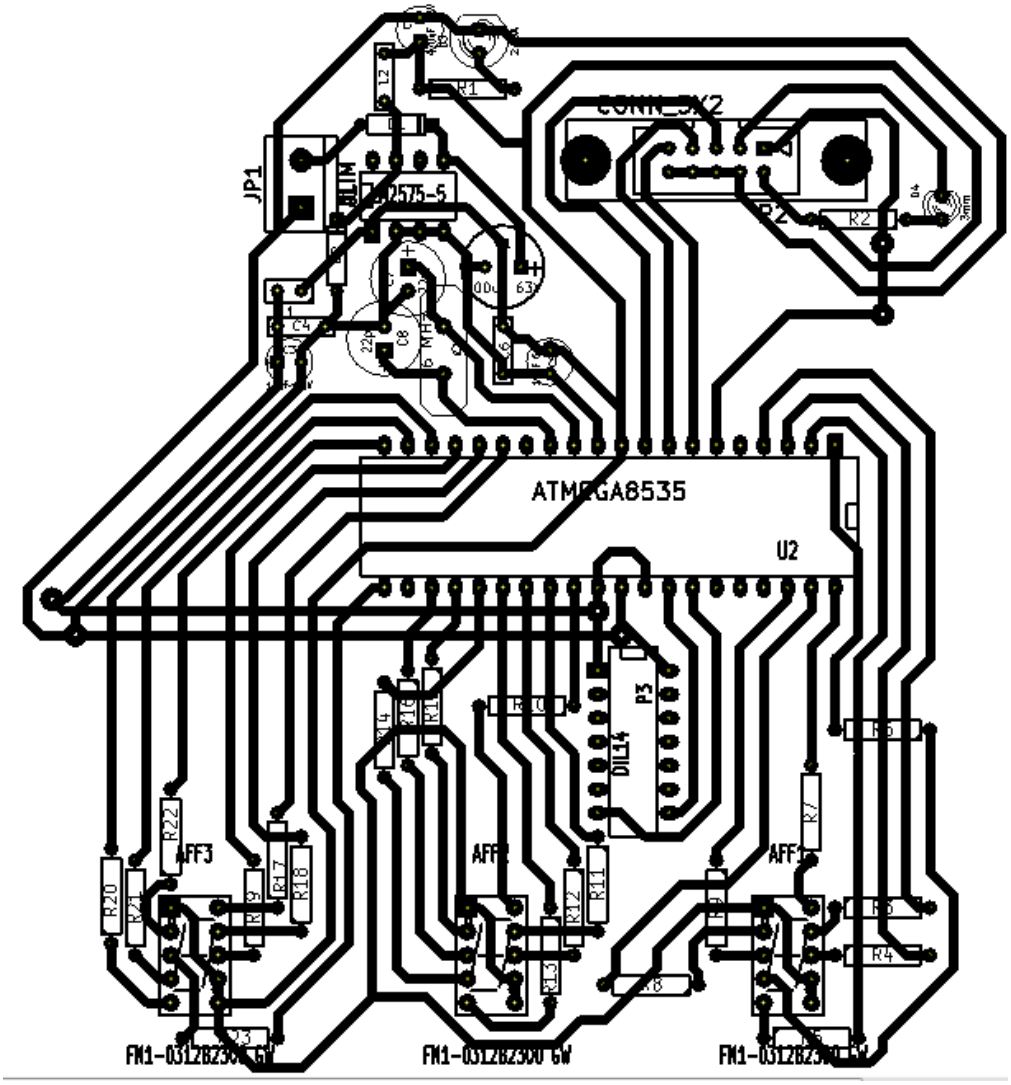


Illustration 20 : carte électronique avec vue de dessus et de dessous une fois routé

Les trois afficheurs se trouvent en bas.

[7] Compatibilité Electromagnétique

Voici le typon qui a permis à graver la carte :

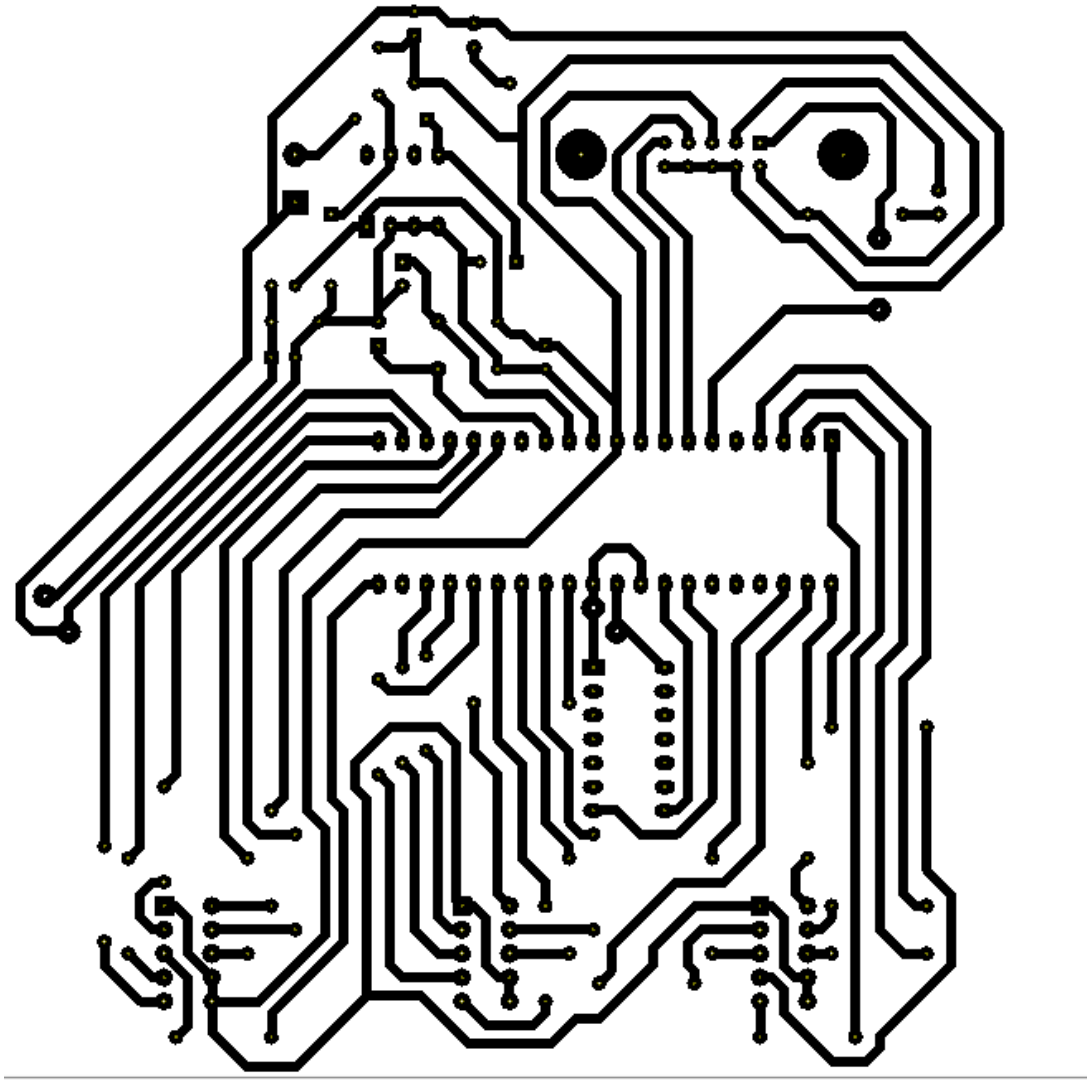


Illustration 21 : typon

Une fois les composants soudés, il ne reste plus qu'à disposer l'accéléromètre sur le connecteur P3.

Maintenant on va pouvoir se lancer dans la programmation du microcontrôleur afin de voir si la carte fonctionne bien.

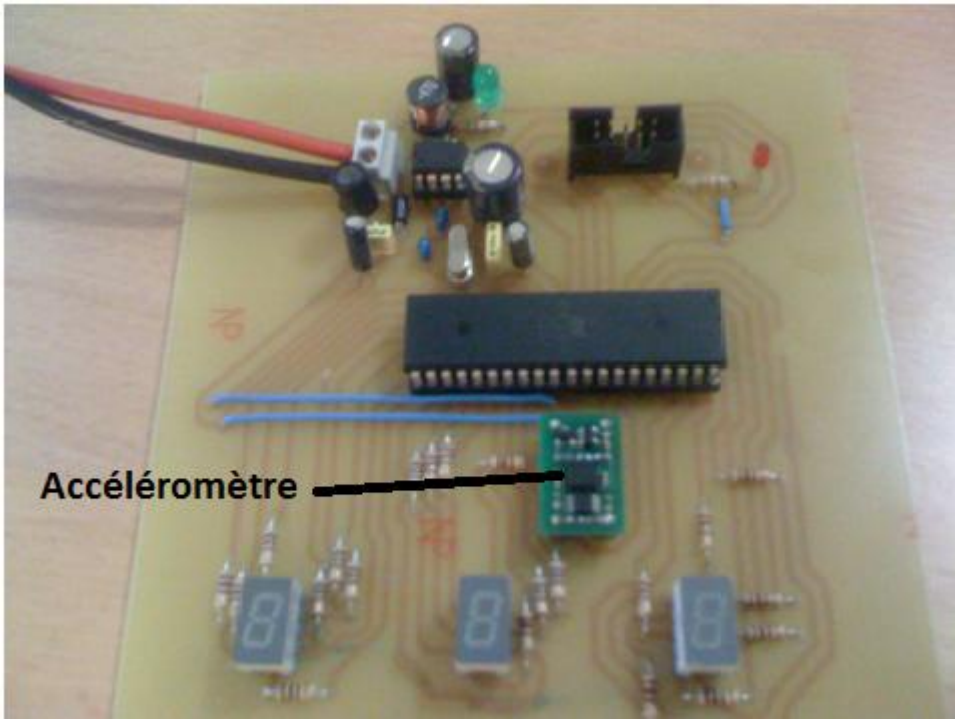


Illustration 22 : Photo de la carte électronique une fois les composants soudés

4. Programmation du microcontrôleur

La programmation de l'ATmega8535 se fait bien entendu sur un ordinateur, dans mon cas avec le logiciel CodeVisionAVR C Compiler Evaluation. C'était la première fois que j'utilisais ce logiciel, au jour où j'écris ce rapport je n'ai pas pu passer beaucoup de temps dessus, mais le logiciel a l'air bien fait et on s'y retrouve vite. En cochant certains paramètres dans le logiciel, selon ce que l'on demande, le logiciel peut produire lui-même plusieurs lignes de codes.

a) Premiers réglages avec le logiciel

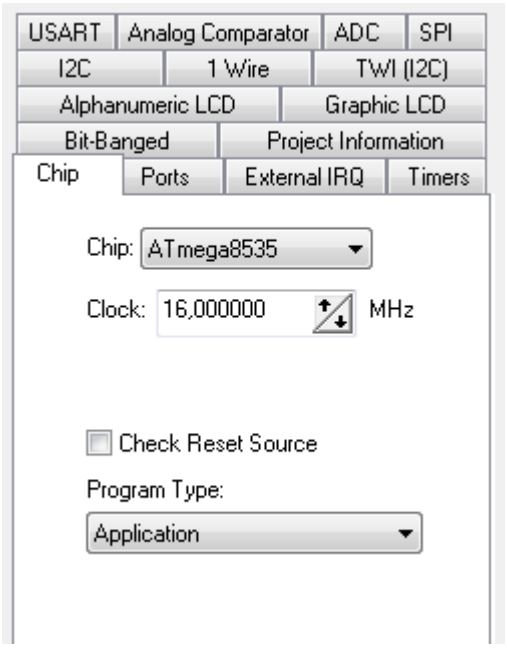


Illustration 23 : Choix du microcontrôleur

Au tout début de la programmation, il y a quelques paramètres à remplir :

Dans l'onglet chip, dans chip on rentre le nom de notre microcontrôleur (ATmega8535)

Et dans clock on met la fréquence de notre horloge, dans notre montage c'est 16 MHz.

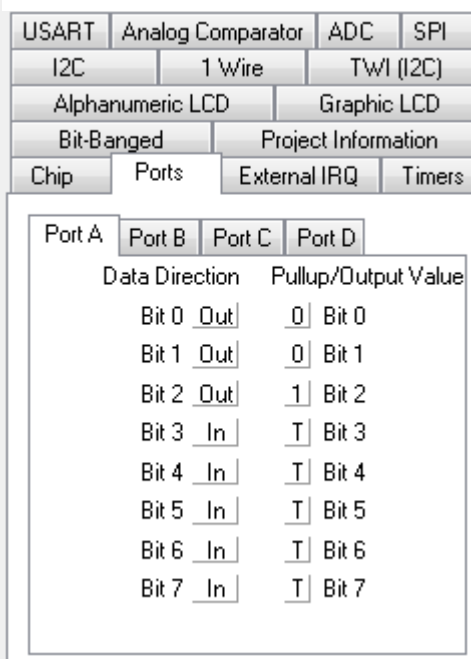


Illustration 24 : Initialisation des 4 ports

Dans l'onglet Ports, on doit configurer l'état initial chaque broche de chaque port en indiquant si c'est une entrée ou une sortie et quelle valeur elle prend. Faire l'opération pour le Port A, B, C et D.

On peut paramétrer beaucoup de choses, comme des timers, des compteurs, les sorties sur écran LCD…

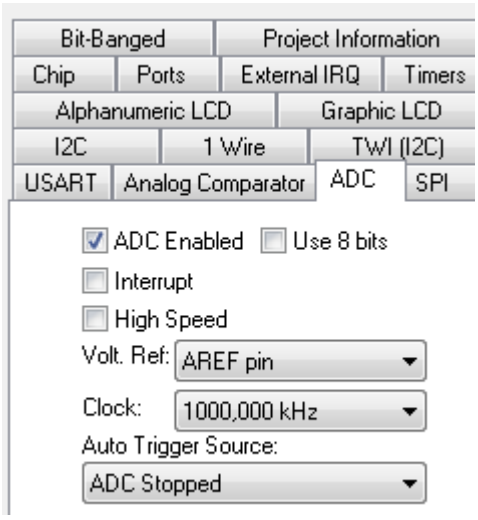


Illustration 25 : Choix d'une conversion Analogique / Numérique

Dans mon cas je me sers de deux entrées établissant la convertion analogique / Numerique.

Il faut que j'aille dans l'onglet ADC et que je coche ADC Enabled.

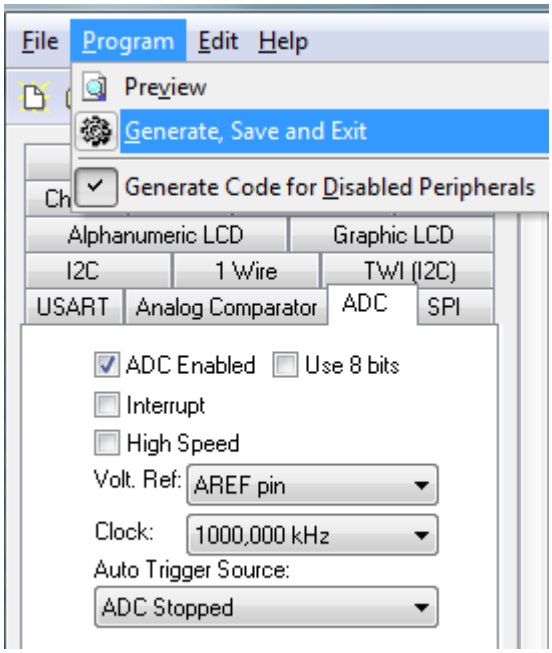


Illustration 26 : Terminer l'entrée des paramètres

Enfin, dès que tous les paramètres sont rentrés, il faut aller dans l'onglet Program / Generate, Save and Exit. A cette étape il faut sauvegarder le fichier puis un programme apparaît, vous pouvez commencer à coder.

b) Compiler et lancer le programme

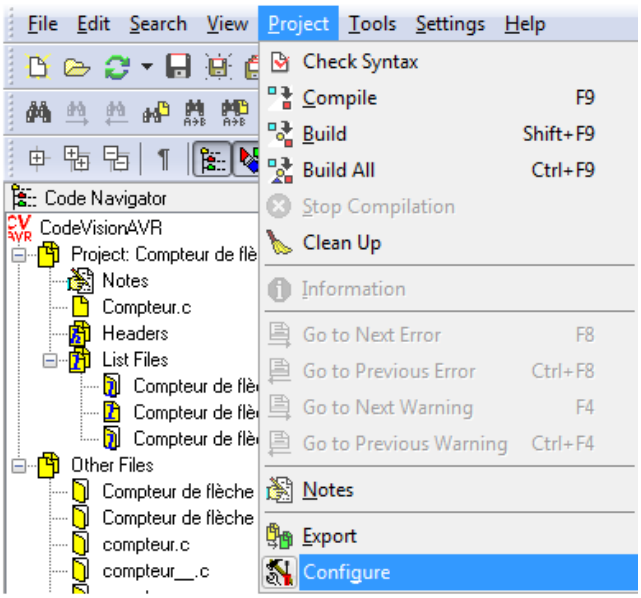
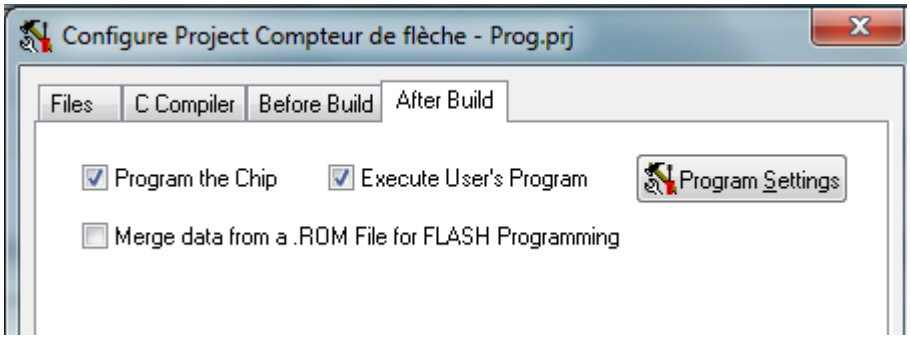


Illustration 27 : Configurer la compilation du programme

Pour lancer l'essai d'un programme, allez dans l'onglet projet / configure.



Il ne faut pas oublier de cocher « Programme the Chip et Execute User's Programme.

Illustration 28 : Critères à cocher

Maintenant si on veut compiler et lancer le projet il suffit d'appuyer sur le bouton « Build all project files » et si il n'y a pas eu d'erreur annoncée, vous pouvez lancer l'envoi du programme dans le Microcontrôleur.

A noter que c'est grâce à la carte visible sur l'illustration 28, que l'on peut transférer notre programme réalisé dans CodeVision.

Cette carte a pour seule utilité de transférer notre programme dans le microcontrôleur pour que celui-ci fonctionne indépendamment d'un ordinateur.

Illustration 29 : Configurer la compilation du programme

c) Programmation : affichage des chiffres sur les afficheurs

Dans un premier temps je veux créer une fonction pour chaque afficheur, permettant à celui–ci d'afficher le nombre qui lui sera rentré en paramètre.

Voici les prototypes de ses trois fonctions :

```
void Aff1( char Nb );
void Aff2( char Nb );
void Aff3( char Nb );
```

Avec Nb la variable rentrée en paramètre, c'est un chiffre allant de 0 à 9 qui sera affiché sur l'afficheur. Je vais utiliser une boucle switch pour affecter un état des bits de sortie du microcontrôleur pour que le chiffre affiché corresponde au chiffre entré en paramètre.

Par exemple si on veut afficher « 0 » sur l'afficheur 2, il faut allumer tous les segments de l'afficheur, sauf celui du milieu : le « G ». Pour cela on enverra des 0 partout sauf pour la broche du segment G. Ce qui donne :

```
PORTC=0b00000010 ;
```

Pour l'afficheur 2, la broche du segment G est connectée au deuxième bit du port C, d'où cette ligne de code.

Une fois les trois fonctions réalisées (voir la dernière annexe avec le programme complet), c'est dans la boucle du while(1) que sont faits tous les traitements car comme la condition du while est 1, le traitement est effectué en continuité. Donc c'est dans cette boucle que je vais venir utiliser les fonctions que j'ai faites en affichant sur les trois afficheurs, tous les chiffres de 0 à 9, histoire de m'assurer que tout ce que j'ai écrit est bon.

d) Programmation : relever une tension analogique

Le fait de relever une tension analogique est fondamental au bon fonctionnement du compteur parce qu'on est obligé de passer par l'étape qui est de relever la tension de sortie de l'accéléromètre puis ensuite de déterminer si cette tension indique qu'une flèche a été tirée.

On s'attend à ce que l'accéléromètre sorte deux tensions allant de 0 à 3V. On utilise un Convertisseur Analogique Numérique pour convertir ces tensions variables, pour cela les tensions sont envoyées dans deux entrées analogiques (broche 6 et 7) qui s'appelle ici ADC6 et ADC7. Ceci permet de générer à partir d'une valeur analogique, une valeur numérique (codée sur plusieurs bits), proportionnelle à la valeur analogique d'entrée. La configuration du CAN :

```
// ADC initialization
// ADC Clock frequency: 125,000 kHz
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x87;
SFIOR&=0xEF;
```

Le convertisseur possède une horloge interne à une fréquence de 0,125 MHz, alimentée sous la tension du microcontrôleur.

La fonction read_adc(1) permet de lire la tension et de la convertir automatiquement. Si la tension en entrée du CAN est de 0V alors en sortie du CAN nous aurons 0 et lorsque la tension est de 5V nous aurons 255 puisque la conversion est faite sur 8 bits. Je vais maintenant faire un traitement qui va permettre de relever la tension sur une des deux sorties de l'accéléromètre et l'afficher sur les afficheurs.

```
char Unit, Diz, Cent, a1, a2, b1, b2;

X = read_adc(7);

a1 = X/10;
a2 = a1*10;
Deci=X-a2;
Aff1(Unit);
```

Déclaration des variables

Lecture et conversion sur 8 bits de la grandeur analogique présente sur la proche 7, puis on range cette valeur dans la variable X

Affichage du chiffre des unités sur l'afficheur 1

```
b1= a1/10;
b2= b1*10;
Diz=a1-b2;
Aff2(Diz);
```

Affichage du chiffre des dizaines sur l'afficheur 2

```
Cent = b1 ;
Aff3(Cent);
```

Affichage du chiffre des centaines sur l'afficheur 3

En testant le programme je me suis aperçu qu'il ne fonctionnait pas comme je voulais, tours les segments restent allumés.

Hélas, arrivant à terme des séances d'E&R je n'ai pas eus le temps de prolonger mes recherches, au profit de la rédaction de ce rapport.

Seulement j'estime que le problème vient, soit des sorties de l'accéléromètre qui pourraient être parasitées par le reste du circuit, soit par mon programme qui est mal construit pour la récupération d'une valeur analogique ou autre hypothèse, (que je pense moins probable) c'est un mauvais traitement de la variable recueillie pour afficher chaque chiffre sur un afficheur.

Pour comprendre d'où vient l'erreur, il me reste à effectuer des recherches sur la sortie de l'accéléromètre, pour voir les valeurs que l'accéléromètre donne à l'ATmega8535 ou même de voir qu'elle valeur, prend la X dans mon programme.

Pour vous expliquer le principe qu'aurait eu le programme informatique : comme les sorties de l'accéléromètre saturent vers les 3 V, on considère que lorsque l'accéléromètre atteindra cette valeur, cela témoignera qu'une flèche à été tirée.

Sachant que l'entrée analogique prendra la valeur numérique de 255 si son entrée analogique est soumise à une tension de 5V. Pour 3V l'entrée analogique relèvera la valeur 153.

Valeur Analogique	Valeur Numérique
5V	255
3V	153
0V	0

Tableau 1 : Tableau de conversion de valeur analogique à numérique

Dans notre cas, le seuil à fixer pour considérer une flèche tirée est de 3V, donc on établira la condition que si la valeur relevée (pour l'instant on a peu d'importance de savoir si c'est l'accélération en X ou en Y que l'on prend, on fera ce choix quand on saura comment sera disposé l'accéléromètre dans le boitier) est plus grande que 150. On prend une valeur légèrement en dessous de 153 pour laisser une petite marge. Et à chaque fois que cette valeur sera dépassée, une variable qui reflétera le nombre de flèches tirées sera complémentée et donc on affichera sur les afficheurs la valeur que prend cette variable.

5. Coût du projet

Voici la liste des prix des composants (à titre indicatif) utilisés pour le projet, si on ne compte pas le détecteur de chocs 801S et l'accéléromètre ACCM2G2. Comme vous pourrez le remarquer, hormis les trois afficheurs et l'accéléromètre, les composants ont été commandés au magasin de l'IUT.

Composant	fournisseur	Valeur	Quantité	Prix unitaire (en €)	Prix total (en €)
Bornier de 2 contacts	magasin IUT		1	0,6	0,6
Connecteur 2x7	magasin IUT		1	0,4	0,4
Connecteur 2x5	magasin IUT		1	0,55	0,55
Résistance	magasin IUT	100Ω	21	0,05	1,05
Résistance	magasin IUT	1,5kΩ	2	0,05	0,1
Diode	magasin IUT	1N4007	1	0,05	0,05
Diode	magasin IUT	1N5819	1	0,25	0,25
LED	magasin IUT	5 mm	1	0,1	0,1
LED	magasin IUT	3 mm	1	0,2	0,2
Condensateur	magasin IUT	100µF 63V	1	0,45	0,45
Condensateur	magasin IUT	470µF 6,3V	1	0,45	0,45
Condensateur	magasin IUT	100nF	2	0,5	1
Condensateur	magasin IUT	10µF 6,3V	2	0,45	0,9
Condensateur	magasin IUT	22pF	2	0,5	1
Quartz	magasin IUT	16 MHz	1	0,6	0,6
Inductence	magasin IUT	47µH	1	0,5	0,5
Inductence	magasin IUT	10µH	1	1,4	1,4
Régulateur	magasin IUT	LM2574-5	1	2,5	2,5
Microcontrôleur	magasin IUT	ATmega8535	1	5	5
Plaque de cuivre	magasin IUT		1	4	4
Afficheur	Fanell	FN1-0312B2300	3	9,5	28,5
Accélérometre	GoTronic	ACCM6G	1	20,9	20,9
TOTAL			47		70,5

Tableau 2 : Liste des composants achetés

6. Planning réel

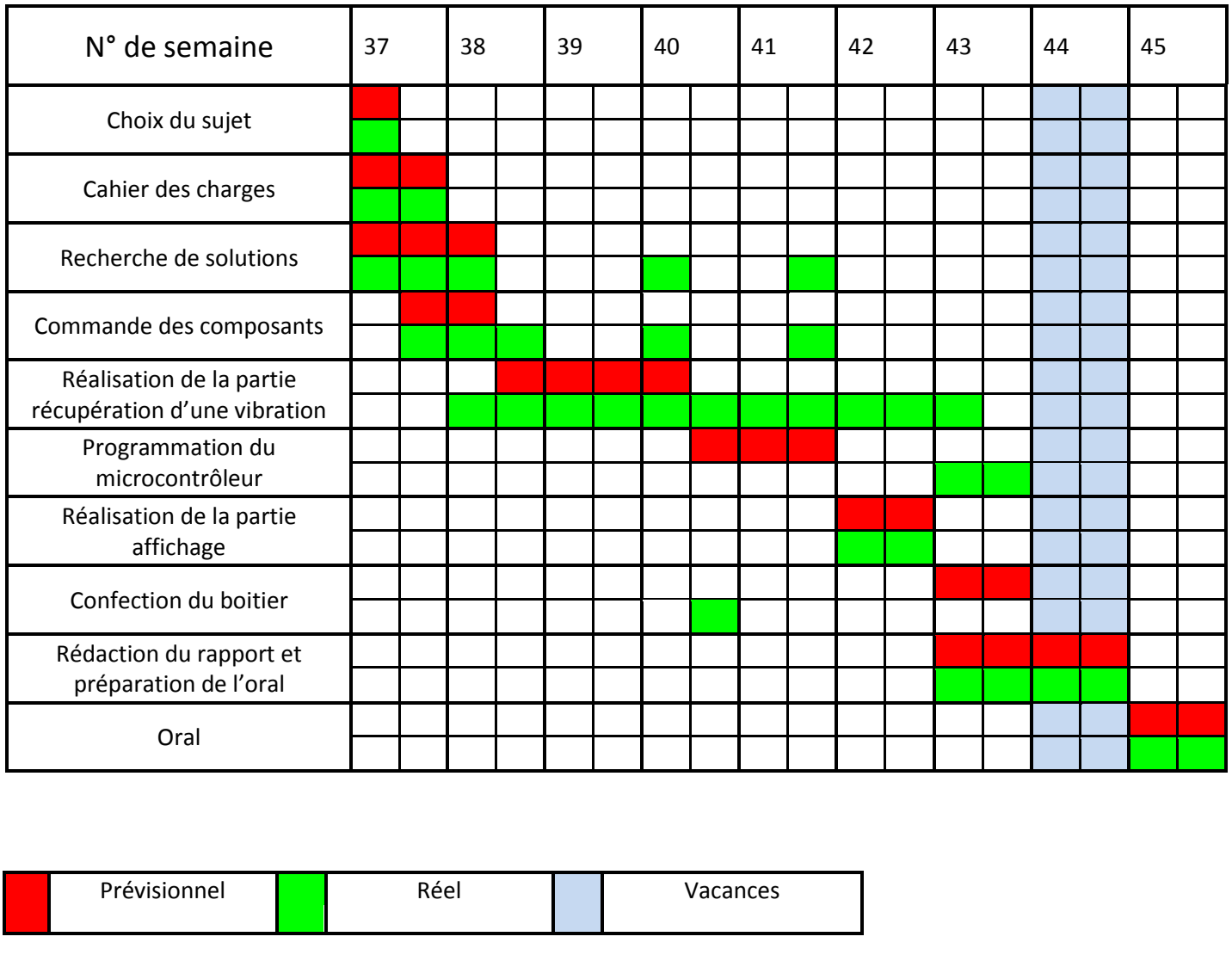


Illustration 30 : Planning réel

Conclusion

Comme vous avez pu le constater, j'ai perdu énormément de temps sur la sélection d'un détecteur de vibrations, c'est le troisième composant testé qui aura été le bon. Seulement en attendant j'ai perdu beaucoup de temps à tester et à comprendre leur fonctionnement, ce qui fait qu'il ne m'est pas resté assez de temps pour faire la partie programmation.

Aujourd'hui, à l'heure de la rédaction de ce rapport, le prototype n'est donc pas opérationnel, je suis déçu de ne pas avoir fini ce projet qui me tenait à cœur, dans les temps, mais je compte bien essayer de trouver la solution ensuite, car je pense qu'il ne manque pas beaucoup de choses afin de faire fonctionner la carte, il faut juste que je cherche à comprendre ce qu'il se passe sur la tension analogique (reflétant l'accélération) lue par l'ATmega8535.

De plus, vous avez sûrement dû vous poser la question, la carte électronique est donc largement trop grosse pour rentrer dans le boitier confectionné. Une fois que la carte avec le programme auront été fonctionnels, on aurait pu se lancer dans la conception d'une carte CMS[8] qui consiste à considérablement réduire l'échelle de la carte, et donc d'avoir une carte miniature qui pourrait rentrer dans le boitier. Autre amélioration c'est un système d'alimentation par pile qui serait intéressant, car le prototype lui est fait pour être alimenté par une tension d'alimentation, alors que les piles rendraient la carte autonome.

Ce projet reste donc à continuer, en espérant avoir du temps pour l'avancer.

8 Composants Montés en Surface

Résumé

J'ai choisi ce projet par ce qu'il me semblait intéressant techniquement avec l'utilisation d'un microcontrôleur ATmega8535 et d'un accéléromètre, que je n'avais jusque là, jamais eu la possibilité d'utiliser et qui sont des composants attractifs par le nombre d'applications pouvant être réalisées. L'objet envisagé aussi était très motivant connaissant l'utilité qu'il aurait dans le domaine du tir à l'arc. C'est seulement au bout du troisième composant que j'ai trouvé le détecteur de vibration qui me convenait. Ce composant sort une tension qui dépend de l'accélération, sachant que l'arc subit une accélération conséquente lors du tir d'une flèche, cette accélération est l'information idéale sur laquelle l'on peut se baser pour compter le nombre de flèches tirées. Cette tension sera donc analysée par l'ATmega8535, qui par l'intermédiaire d'un programme rentré et l'utilisation de la fonction read_adc() comptera qu'une flèche à été tirée lorsque la tension envoyé par l'accéléromètre dépassera une certaine valeur. C'est ici que mon projet échoue et empêche le fonctionnement souhaité. Une fois l'information reçue, une variable est complémentée dans le microcontrôleur puis, grâce à ses ports de sortie, affichée sur les afficheurs afin que l'archer puisse savoir le nombre de flèches qu'il a tirées.

Bibliographie

Gotronic. (s.d.). Consulté le 09 2013, sur Accélérométres: infos et exemples d'utilisation: http://www.gotronic.fr/pj-36.pdf

LEQUEU, T. (2013, 09 19). *Programmation ATmega8535.* Consulté le 10 2013, sur La documentation de Thierry LEQUEU sur OVH: http://www.thierry-lequeu.fr/data/DIV517.HTM

SAURET, F. (2005, Juin 24). *téléchargement.* Consulté le 10 2013, sur AnyEDIT.fr: http://anyedit.free.fr/telechargement/atmega8535-francais.pdf

Mots clefs

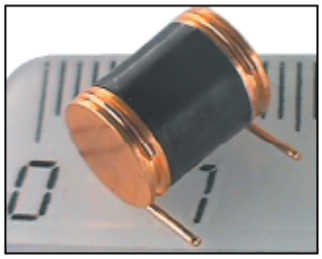
- Compteur
- Vibration
- Accélération
- Accéléromètre
- Détecteur de chocs
- Microcontrôleur
- Alimentation
- Programme
- Variable
- Logiciel
- Tension
- Afficheur
- LED
- Segment
- Axe
- Résistance
- Condensateur
- Inductance
- LED
- Broche
- Port
- Carte électronique
- Typon
- Boitier

Tables des illustrations

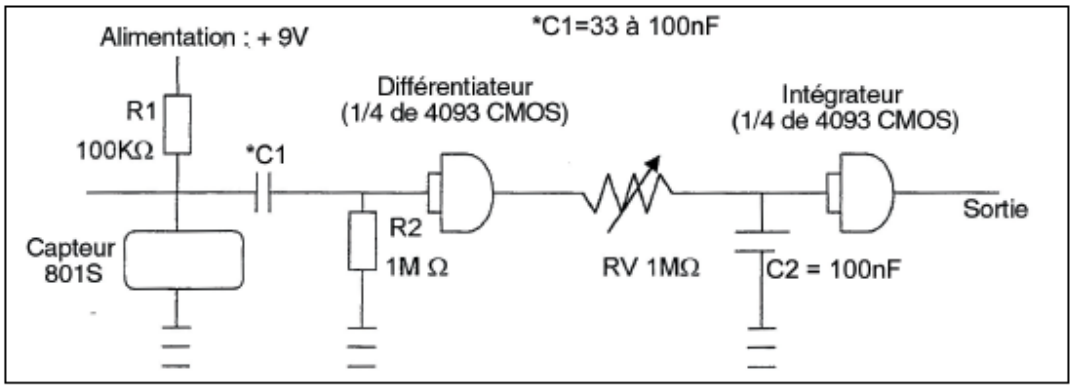
Annexe 1

DÉTECTEUR DE CHOCS 801S référence : 9344

- Détection de chocs de très faible amplitude
- Détection dans toutes les directions
- Détecte également les changements de position
- Durée de vie garantie d'au moins 60.000.000 de détections
- Circuit à faible coût permettant d'ajuster la sensibilité de détection
- Fonctionne sans mercure
- Très grande sensibilité
- Entièrement statique
- Pour systèmes d'alarme, etc
- Dimensions : Ø 7 x 9,2 mm
- Sorties picots Ø 0,5 mm pour circuit imprimé

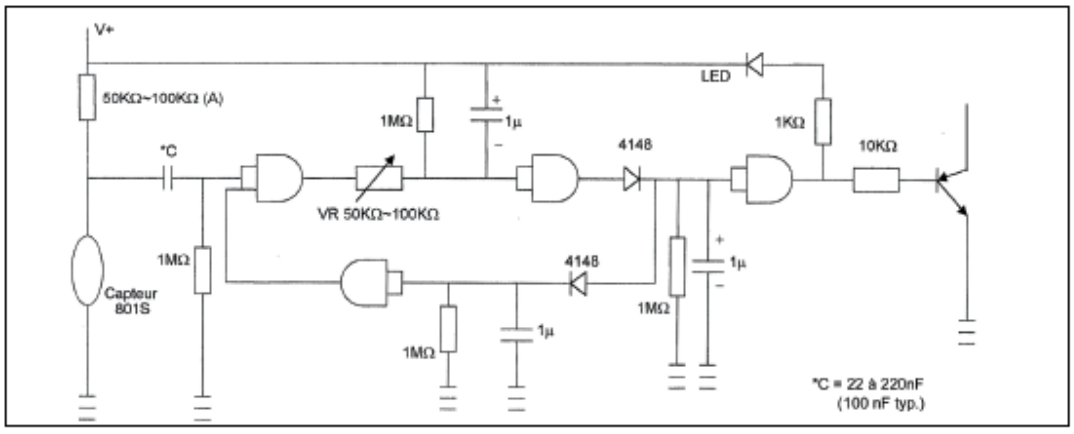


EXEMPLES D'APPLICATIONS CIRCUIT DE DÉTECTION SIMPLE



ATTENTION :
La sortie de ce montage est un signal rectangulaire en présence de détection et non un niveau haut ou bas stable.

CIRCUIT DE DÉTECTION DE CHOCS POUR ALARME AUTOMOBILE



REMARQUES :

1. Le circuit intégré est un CD4093 CMOS.
2. C = 22 à 220 nF (100 nF typ.)
3. Alimentation du montage : 12 VDC
4. La sensibilité peut être ajustée en jouant sur les valeurs de A et B dans les limites indiquées.

Importé et distribué par :

Selectronic • 86, rue de Cambrai • 59000 LILLE
Tél.: (0) 328.550.328 • FAX : (0) 328.550.329
www.selectronic.fr

Annexe 2

DimensionEngineering

DE-ACCM2G2 Buffered ±2g Accelerometer

General Description
The DE-ACCM2G2 is an off the shelf 2 axis 2g accelerometer solution with analog outputs. It features integrated op amp buffers for direct connection to a microcontroller's analog inputs, or for driving heavier loads.

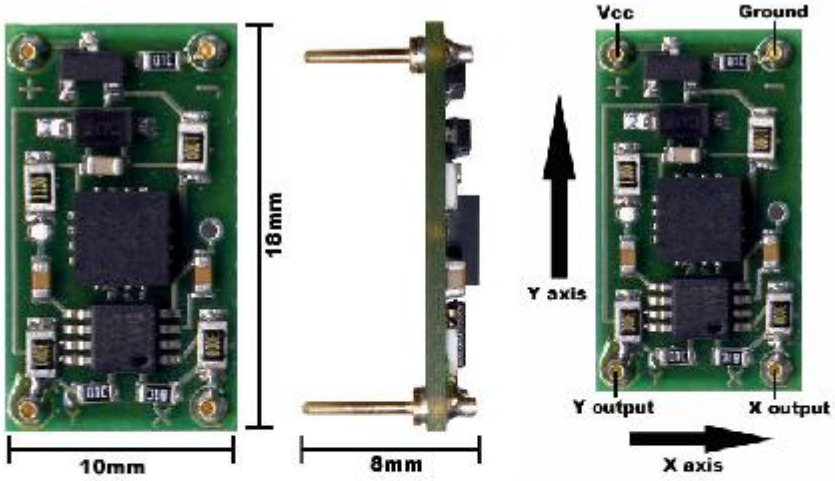
Additional circuitry ensures that the product won't be damaged by reversed power connections, or voltages above the recommended ratings.

The DE-ACCM2G2 is designed to fit the DIP-14 form factor, making it suitable for breadboarding, perfboarding, and insertion into standard chip sockets.

It is based on the ST MicroElectronics LIS244ALH for superior sensitivity and lower cost.

Features
Dual axis ±2g sense range
660mV/g sensitivity
500Hz bandwidth
Operating voltage 3.5V to 15V (onboard regulator)
Reverse voltage protection
Overvoltage protected up to 14V
Output short protection
Standard DIP-14 form factor
Integrated power supply decoupling
Draws under 2mA
<4% typical 0g bias deviation from Vcc/2
Shiny gold pins to distract the enemy!

Applications
Motion, tilt and slope measurement
Shock sensing
Vehicle acceleration logging



Annexe 3

(Ce qui est écrit après « // » correspond à du commentaire et nom du code)

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.2c Professional
Automatic Program Generator
© Copyright 1998-2004 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.ro
e-mail:office@hpinfotech.ro

Project :
Version :
Date    : 23/10/2013
Author  : F4CG
Company : F4CG
Comments:

Chip type           : ATmega8535
Program type        : Application
Clock frequency     : 16,000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 128
*****************************************************/

#include <mega8535.h>
#include <delay.h>  // Pour avoir la fonction delay_ms().

#define ADC_VREF_TYPE 0x00

#define Aff1A PORTA.0      //Définition des broches allant à l'afficheur1
#define Aff1B PORTB.3
#define Aff1C PORTB.2
#define Aff1D PORTB.1
#define Aff1E PORTB.0
#define Aff1F PORTA.1
#define Aff1G PORTA.2
#define Aff1Alim PORTB.4
#define Aff2A PORTC.7      //Définition des broches allant à l'afficheur2
#define Aff2B PORTC.6
#define Aff2C PORTC.5
#define Aff2D PORTC.4
#define Aff2E PORTC.3
#define Aff2F PORTC.2
#define Aff2G PORTC.1
#define Aff2Alim PORTC.0
#define Aff3A PORTD.1      //Définition des broches allant à l'afficheur3
#define Aff3B PORTD.2
#define Aff3C PORTD.3
#define Aff3D PORTD.4
```

```
#define Aff3E PORTD.5
#define Aff3F PORTD.6
#define Aff3G PORTD.7
#define Aff3Alim PORTD.0

#define Y PORTA.6        // Sortie X de l'accéléromètre
#define X PORTA.7        // Sortie Y de l'accéléromètre

unsigned int X;          // déclaration de la variable X en global

void Aff1( char Nb );    // prototypes des fonction servant à afficher
void Aff2( char Nb );
void Aff3( char Nb );

void Aff1( char Nb)      // fonction servant à afficher un chiffre sur l'afficheur 1
{
        switch( Nb )
        {
                case 0:             // écriture du chiffre 0 sur l'afficheur
                PORTA=0b00000100;   // on met à 1 les broches de l'afficheur reliés
                PORTB=0b00000000;   // à un segment que l'on ne veut pas allumer
                break;

                case 1:
                PORTA=0b00000111;
                PORTB=0b00000011;
                break;

                case 2:
                PORTA=0b00000010;
                PORTB=0b00000100;
                break;

                case 3:
                PORTA=0b00000010;
                PORTB=0b00000001;
                break;

                case 4:
                PORTA=0b00000001;
                PORTB=0b00000011;
                break;

                case 5:
                PORTA=0b00000000;
                PORTB=0b00001001;
                break;

                case 6:
                PORTA=0b00000000;
                PORTB=0b00001000;
                break;

                case 7:
                PORTA=0b00000110;
```

```
                PORTB=0b00000011;
                break;

                case 8:
                PORTA=0b00000000;
                PORTB=0b00000000;
                break;

            case 9:
                PORTA=0b00000000;
                PORTB=0b00000001;
                break;

        }
}

void Aff2( char Nb)       // fonction servant à afficher un chiffre sur l'afficheur 2
{
        switch( Nb )
        {
                case 0:
                PORTC=0b00000010;
                break;

                case 1:
                PORTC=0b10011110;
                break;

                case 2:
                PORTC=0b00100100;
                break;

                case 3:
                PORTC=0b00001100;
                break;

                case 4:
                PORTC=0b10011000;
                break;

                case 5:
                PORTC=0b01001000;
                break;

                case 6:
                PORTC=0b01000000;
                break;

                case 7:
                PORTC=0b00011110;
                break;

                case 8:
                PORTC=0b00000000;
                break;
```

```
            case 9:
                PORTC=0b00001000;
                break;

        }
}

void Aff3( char Nb)      // fonction servant à afficher un chiffre sur l'afficheur 3
{
        switch( Nb )
        {
                case 0:
                PORTD=0b10000000;
                break;

                case 1:
                PORTD=0b11110010;
                break;

                case 2:
                PORTD=0b01001000;
                break;

                case 3:
                PORTD=0b01100000;
                break;

                case 4:
                PORTD=0b00110010;
                break;

                case 5:
                PORTD=0b00100100;
                break;

                case 6:
                PORTD=0b00000100;
                break;

                case 7:
                PORTD=0b11110000;
                break;

                case 8:
                PORTD=0b00000000;
                break;

            case 9:
                PORTD=0b00100000;
                break;
        }
}
```

```
// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=Out Func1=Out Func0=Out
// State7=T State6=T State5=T State4=T State3=T State2=1 State1=0 State0=0
PORTA=0x04;
DDRA=0x07;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=T State6=T State5=T State4=0 State3=0 State2=0 State1=0 State0=0
PORTB=0x00;
DDRB=0x1F;

// Port C initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=1 State0=0
PORTC=0x02;
DDRC=0xFF;

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=1 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTD=0x80;
DDRD=0xFF;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
```

```
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 125,000 kHz
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x87;
SFIOR&=0xEF;


while (1)
```

```
    {
    // Place your code here
    char Deci, Diz, Cent, a1, a2, b1, b2;

    X = read_adc(7);
    a1 = X/10;       // on divise par 10 la valeur lue
    a2 = a1*10;      // on la multiplie par 10, ce qui nous permet de mettre à 0 le chiffre des unités
    Deci=X-a2;       // le chiffre à afficher est la différence entre la valeur relevé et le nombre calculé
    Aff1(Deci);      // on affiche le chiffre sur l'afficheur 1 grâce à la fonction Aff1()

    b1= a1/10;       // même exercice pour afficher le chiffre des dizaines
    b2= b1*10;
    Diz=a1-b2;
    Aff2(Diz);

    Cent = b1 ;     // le chiffre des centaines correspond au chiffre entier de la valeur lue
    Aff3(Cent);     // divisée par 100, on ne prend pas en compte les chiffre après la virgule

    };
}
```