REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT

SUPERIEUR ET DE LA RECHERCHE

SCIENTIFIQUE

UNIVERSITE AMAR TELIDJI - LAGHOUAT

Faculté de Technologie

Département d'Electronique

MEMOIRE DE MASTER

Présenté par : Gueddouda Israa

DOMAINE : Sciences et Techniques

FILIERE : Electronique

OPTION : Instrumentation

Thème

Proposition d'un programme qui détermine les déplacements d'une souris optique à partir des données brutes, alternative au programme intégré dans cette souris

Jury de soutenance :

CHOUIREB FATIMA	**Pr**	**Président**
BIRANE ABDELKADER	**MCB**	**Examinateur**
FEKNOUS SAFIA	**MAA**	**Rapporteur**

Promotion : 2021-2022

Résumé

Une souris optique est constituée d'un capteur d'image, un système optique d'éclairage, et une électronique et programme de traitement des données. On s'intéresse dans ce projet seulement à la fonction de mesure des déplacements de la souris. L'objectif de ce projet est de proposer des programmes de gestion de la souris, alternatifs aux programmes résidents dans la souris en se basant seulement sur notre connaissance de son principe de fonctionnement. L'un des programmes proposés fonctionnera sur une carte Arduino UNO et aura pour fonction la communication entre la carte Arduino et le capteur de la souris d'une part, et la communication entre la carte Arduino et le PC d'autre part. Le deuxième programme fonctionne sur le PC et a pour tâche de faire des traitements des images capturées basés sur la corrélation pour en déduire les déplacements de la souris. La communication entre l'Arduino et le PC doit permettre de fournir les images capturées par le capteur au programme de traitement. Les essais ont été faits et on donnés des résultats satisfaisant. Seul le moyen qu'on avait prévu d'utiliser pour la communication entre le programme de traitement sur le PC et la carte Arduino, qui est la bibliothèque SerialPort n'a pas bien fonctionné, et on a pas eu le temps de chercher un autre moyen. A défaut d'un moyen de communication entre le PC et la carte Arduino on s'est contenter d'un traitement offline des images capturées.

Mots clés :

Souris optique – Capteur ADNS-5050 - Carte Arduino – Traitement de corrélation d'images – Bibliothèque OpenCV - Bibliothèque de communication série entre Arduino et PC.

ملخص

يتكون الفأرة الضوئية من مستشعر الصورة ونظام الإضاءة الضوئية وبرنامج معالجة الإلكترونيات والبيانات. في هذا المشروع ، نحن مهتمون فقط بوظيفة قياس حركات الماوس. الهدف من هذا المشروع هو اقتراح برامج إدارة الماوس ، كبديل للبرامج المقيمة في الماوس ، بناءً على معرفتنا بمبدأ التشغيل الخاص بها فقط. سيعمل أحد البرامج المقترحة على لوحة الاردوينواونو وستكون لها وظيفة الاتصال بين لوحة الاردوينو ومستشعر الماوس من ناحية ، والاتصال بين لوحة الاردوينو والكمبيوتر الشخصي من ناحية أخرى. يعمل البرنامج الثاني على جهاز الكمبيوتر وتتمثل مهمته في معالجة الصور الملتقطة بناءً على الارتباط لاستنتاج حركات الماوس. يجب أن يتيح الاتصال بين الاردوينو والكمبيوتر الشخصي توفير الصور الملتقطة بواسطة المستشعر لبرنامج المعالجة. تم إجراء الاختبارات وحققت نتائج مرضية. فقط الوسائل التي خططنا لاستخدامها للاتصال بين برنامج المعالجة على الكمبيوتر ولوحة الاردوينو، وهي مكتبة سريال بورت، لم تعمل بشكل جيد ، ولم يكن لدينا الوقت للبحث عن وسيط آخر. في حالة عدم وجود وسيلة اتصال بين الكمبيوتر ولوحة الاردوينو، قررنا معالجة الصور الملتقطة في وضع عدم الاتصال.

الكلمات المفتاحية :

الماوس البصري - مستشعر ADNS-5050 - لوحة Arduino - معالجة ارتباط الصور - مكتبة OpenCV - مكتبة الاتصالات التسلسلية بين Arduino والكمبيوتر الشخصي.

Abstract

An optical mouse consists of an image sensor, an optical lighting system, and an electronics and data processing program. In this project, we are only interested by measuring the mouse movements. The objective of this project is to propose mouse management programs, alternative to resident programs in the mouse, based only on our knowledge of its operating principle. One of the proposed programs will work on an Arduino UNO board and will have the function of communication between the Arduino board and the mouse sensor on the one hand, and the communication between the Arduino board and the PC on the other hand. The second program runs on the PC and has the task of processing the captured images based on the correlation to deduce the mouse movements. The communication between the Arduino and the PC must make it possible to provide the images captured by the sensor to the processing program. The tests have been carried out and have given satisfactory results. Only the means we had planned to use for communication between the processing program on the PC and the Arduino board, which is the SerialPort library, did not work well, and we did not have time to look for another medium. In the absence of a means of communication between the PC and the Arduino board, we settled for offline processing of the captured images.

Keywords :

Optical mouse - ADNS-5050 sensor - Arduino board - Image correlation processing - OpenCV library - Serial communication library between Arduino and PC.

Remerciements

J’offre ma grande gratitude à Dieu qui m’a aidé à faire ce travail.

J’exprime ma profonde gratitude à mes parents pour leurs encouragements, leurs soutiens et pour les sacrifices qu’ils ont enduré.

Je remercie ma promotrice ***Mme FEKNOUS Safia*** *pour ses efforts qu’elle a déployé, pour m’aider, conseiller, encourager et me corriger.*

Et je remercie le Chef département ***Mr. MERRAH Lahcen*** *de m'avoir aidé dans mon projet en nous procurant des anciennes souris avec capteurs d’Avago Technologies*

Je voudrais remercier les membres de jury d’avoir accepté d’examiner mon travail.

Je remercie aussi tout le corps enseignant dans le département d'électronique qui a contribué à ma formation universitaire.

Sans oublier tous mes amis.

Que tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail trouvent ici ma sincère reconnaissance

Sommaire

Chapitre III : Programmes et résultats

Introduction

Une souris optique est un système embarqué constitué d'une électronique et des programmes construits suivant un certain principe de base. L'objectif qu'on s'est donné dans ce projet part d'une question : « Serions-nous capable de construire notre propre **programme** qui gère cette souris connaissant seulement son principe de fonctionnement ? »

Bien sure, notre but principal n'est pas de trouver programme qui permet d'avoir un fonctionnement optimal de la souris optique. Mais le but est plutôt de proposer un programme de fonctionnement **opérationnels** même avec des performances de fonctionnement moyens. Car le fait de proposer un programmes de fonctionnement pour souris optique est en lui-même un défit. C'est dans ce contexte qu'on est partit explorer et découvrir cet appareil.

La fonction sur laquelle on s'est focalisé est celle de la mesure des déplacements de la souris. On ne s'intéressera ni aux fonctions de boutons gauche et droite, ni à la roulette de défilement.

Pour pouvoir aborder ce travail, on devait bien comprendre son architecture et son principe de fonctionnement et le principe de fonctionnement du capteur d'images et son DSP. Ceci n'aurait pas été possible sans une souris optique avec un capteur particulier, celui du constructeur Avago (Agilent) dont le datasheet fournit des informations détaillées sur son fonctionnement. Seulement, on ne peut trouver de tels capteurs que dans les anciennes souris, parce qu'Avago ne produit plus de capteurs de souris optiques. Les datasheet des capteurs utilisés dans les souris actuelles ne fournissent pas des informations pertinentes telles que l'accès aux registres internes du DSP.

On rencontre sur internet quelques projets individuels avec des souris optiques. C'est ce qui nous a poussé et encouragé à tenter l'expérience nous aussi. Et surtout, on voit que l'expérience est réalisable, puisque d'autres personnes on réussit à exploiter les souris optiques et on obtenu des résultats impressionnants. Certains emploient le terme « hacker » une souris optique. Parmi les projets qu'on rencontre, il y a l'affichage des images capturées par le capteur d'une souris optique, la réalisation d'un scanner avec une souris optique, la réalisation d'un capteur de position pour robot mobile avec une souris optique, etc.

Une souris optique détermine ses déplacements grâce aux traitements faits par son DSP sur les images capturées de la surface de travail. On n'a pas encore rencontré de travail qui propose d'écrire un programme qui traite lui-même les images capturées alternatif au programme intégré dans le DSP.

Ce mémoire est divisé en trois chapitres. Le premier chapitre présente la souris d'ordinateur de façon général et son évolution au cours des années passés. Ensuite on parlera en particulier de la souris optique : ses constituants, son capteur, les traitements qui détermine les déplacements de la souris. Ensuite on donne les détails sur le fonctionnement du capteur ADN-5050 qu'on a utilisé dans notre projet.

Le chapitre II présente les moyens matériels et logiciels qu'on a utilisé dans ce projet. Après avoir présenté le schéma complet du système réalisé, on présente les parties matériels (la souris optique et la carte Arduino UNO) et logiciel (Bibliothèque Arduino pour capteur ADNS-5050, programme de corrélation, et bibliothèque pour la communication série entre l'Arduino et programme qui marche sur le PC) qui le constitue.

Le chapitre III présente le travail fait (circuits électroniques et programmes) et les résultats obtenus. Et on termine enfin par une conclusion.

Chapitre I
La souris optique

I.1 Définition d'une souris d'informatique

Une souris informatique (ou souris d'ordinateur) est un dispositif de pointage qu'on relie à un ordinateur avec ou sans fil. La souris permet à l'utilisateur d'agir sur **l'interface utilisateur graphique** affiché sur l'écran de l'ordinateur en sélectionnant certains objets de cette interface au moyen de clics sur les boutons de la souris. Avant cela, on doit d'abord ramener l'élément de l'interface graphique appelé « **curseur** » (figure I.1) sur l'objet graphique à sélectionner par déplacement de la souris sur la surface de travail (bureau, table, …).

Figure I.1 Souris informatique et pointeur de souris

I.2 Historique et évolution des souris d'ordinateurs

Avec l'introduction de l'interface utilisateur graphique (GUI : Graphical User Interface) dans les systèmes d'ordinateur, la souris est devenue un outil indispensable.
Le nom donné à la première souris était "Indicateur de position X-Y pour système d'affichage". Pour faire plus léger on lui donna plus-tard le nom de « souris » à cause de son apparence similaire à cet animal.

I.2.1 Première souris d'ordinateur

Elle a été inventée en 1963 par Douglas Englebart. De l'extérieur, elle était composée d'une coque en bois, un bouton, deux disques en dessous, perpendiculaire l'un par rapport à l'autre, et un câble torsadé. Son utilisation consiste à faire rouler les deux disques sur la surface du bureau pour déplacer le curseur sur l'écran. Son mécanisme est très simple. Chaque disque fait tourner un potentiomètre (figure I.2 droite) dont la tension est envoyé à l'ordinateur à travers le câble pour mesurer le mouvement de la souris suivant les axes X et Y. L'état du bouton de la souris est aussi envoyé à travers ce même câble. Ainsi, on peut déplacer le curseur sur l'écran de la même façon que la souris **[w1]**. Cette souris n'était néanmoins pas très pratique car elle était limitée dans son mouvement, car elle ne pouvait se déplacer que suivant un seul axe, horizontal ou vertical. Les souris d'aujourd'hui ont des mouvements plus fluides puisqu'elles peuvent se déplacer dans n'importe quelle direction.

Figure I.2 Inventeur de la première souris d'ordinateur (**à gauche**) - Première souris d'ordinateur (**au milieu**) - Schéma de principe de la première souris d'ordinateur (**à droite**)

I.2.3 Le trackball

En réalité la souris informatique a été inventé bien avant 1968, mais étant un projet militaire cette invention a été gardée secrète. Le trackball appelé aussi « boule roulante » (rolling ball), est un dispositif de pointage inventé en 1946 par **Ralph Benjamin** qui travaillait alors pour le service scientifique de la marine royale britannique dans le cadre d'un système de traçage radar de l'après-Seconde Guerre mondiale. Un système capable de calculer la trajectoire théorique des aéronefs surveillés en fonction des entrées d'un utilisateur (figure I.3.b). Dans l'appareil de Benjamin, la main déplace la balle elle-même. Il s'agissait donc d'une grande souris mécanique fixe, à l'envers. Seul un prototype utilisant une boule de métal roulant sur deux roues recouvertes de caoutchouc a été a été breveté en 1947 (figure I.3). Par contre le mécanisme complet de la souris qui permettait de déplacer le curseur sur l'écran a été gardé comme un secret militaire. De nos jours le système de trackball est employé sur certains modèles de souris modernes (figure I.3 droite).

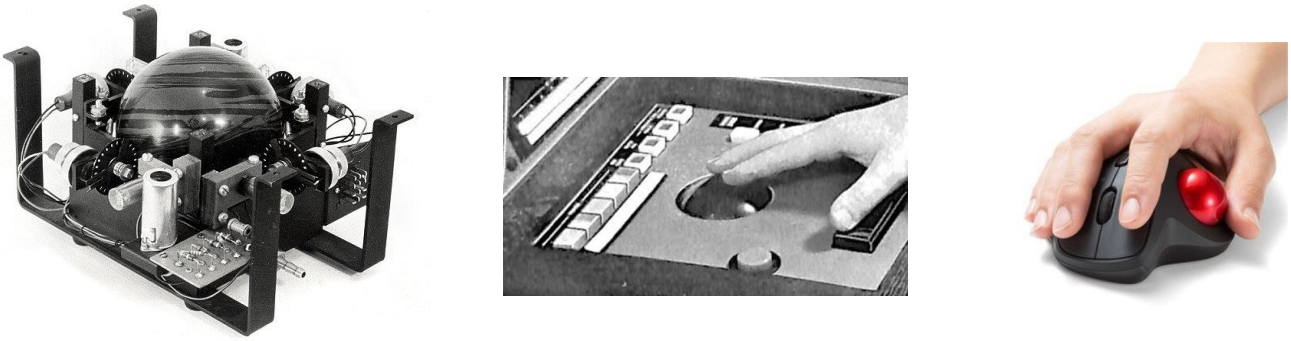


Figure I.3 Première souris track-ball (**à gauche**) **–** Ancien appareil avec souris track-ball (**au milieu**) – Souris moderne avec track-ball (**à droite**)

I.2.3 Souris informatique à boule

En raison de sa conception primitive et les restrictions de mouvement de la première souris informatiques, on a commencé à développer des souris omnidirectionnelles qui peuvent se déplacer dans n'importe quelle direction sur le plan de travail. C'est ainsi que la souris à boule a vu le jour. En 1968, la société allemande **Telefunken** lance le Rollkugel (qui veut dire "balle qui roule"). Cette souris n'était en réalité qu'un trackball inversé. La friction de la boule contre la table permettait le mouvement du pointeur sur l'écran.

Figure I.4 Première souris à boule et son inventeur

Cette souris à boule a ensuite été améliorée en remplaçant le capteur de position potentiométrique par un encodeur optique **[w1]**. Souris qu'on qualifie d'optomécaniques.

Une souris à boule typique est principalement constituée d'une boule, deux encodeurs optiques, un processeur et son microprogramme (firmware) qui calcule le déplacement x, y en fonction des signaux d'impulsions générées par les deux encodeurs (figure I.6 gauche et milieu).

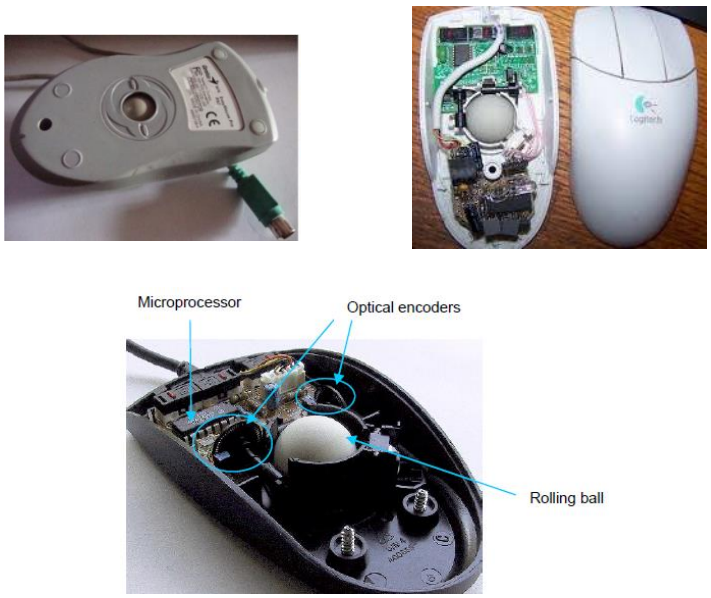


Figure I.5 Souris à boule vue de l'extérieur et de l'intérieur

Quand on déplace la souris sur la table, la boule tourne et transmet son mouvement par frottement à deux axes perpendiculaires qui portent chacun deux disques avec un certain nombre de trous équidistants sur leurs bords. Chacun de ces disques forme avec une paire de LED et une paire de photodiodes placées de part et d'autre du disque, l'encodeur optique qui sert à mesurer les déplacements de la souris.

Chaque LED est placée en face d'une photodiode qui reçoit la lumière de celle-ci au passage d'un trou du disque quand ce dernier tourne, et ne reçoit rien quand il n'y a pas de trou. Quand la souris se déplace, on obtient à la sortie de la photodiode, un signal électrique d'impulsions (après mise en forme du signal). Pour chaque axe, les LED et les photodiodes sont placées de façon à obtenir deux signaux électriques d'impulsions en quadrature de phase (déphasage de 90°) (figure I.7.b).

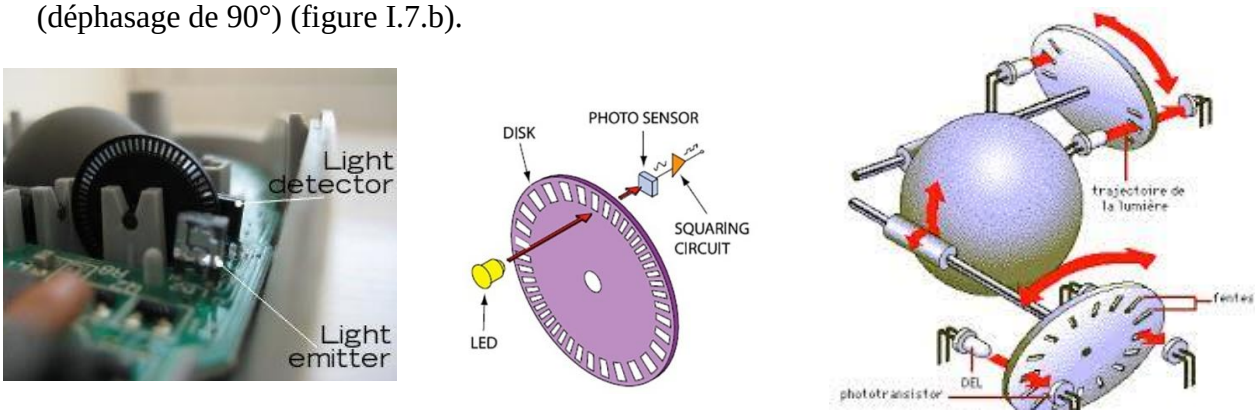


Figure I.6 Encodeur optique dans une souris à boule (**à gauche**) – Constituants d'un encodeur optique (**au milieu**) – Mécanisme d'une souris à boule (**à droite**)

Le processeur mesure des déplacements de la souris suivant les axes X et Y par comptage des impulsions. L'information du sens des déplacements suivant X et Y (en avant ou en arrière, à gauche ou à droite) réside dans le déphasage entre les signaux provenant des deux photodiodes, qui peut être soit +90° soit -90° (figure I.7.c). Pour déterminer facilement ce déphasage, le processeur génère à partir de ces deux signaux déphasés, des valeurs binaires en code de Gray ascendant ou descendant (figure 1.7.c et Tableaux I.1).

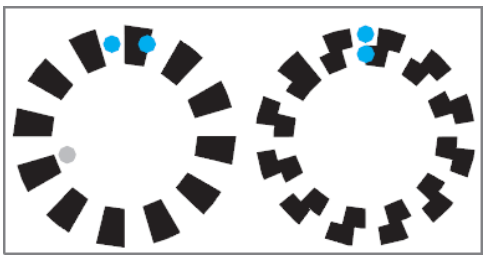


Figure I.7.a Deux types d'encodeurs : Paire d'E/R IR déphasées et un disque troué (à gauche) Une paire d'E/R IR en phase et deux disques troués déphasés

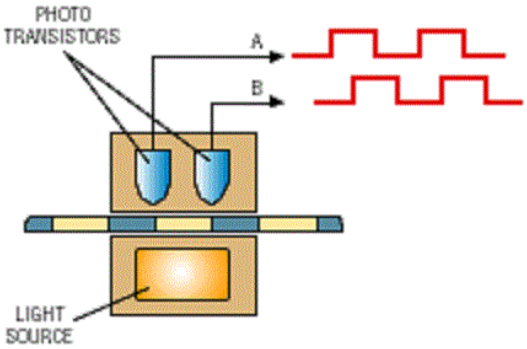


Figure I.7.b Signaux électriques de l'encodeur

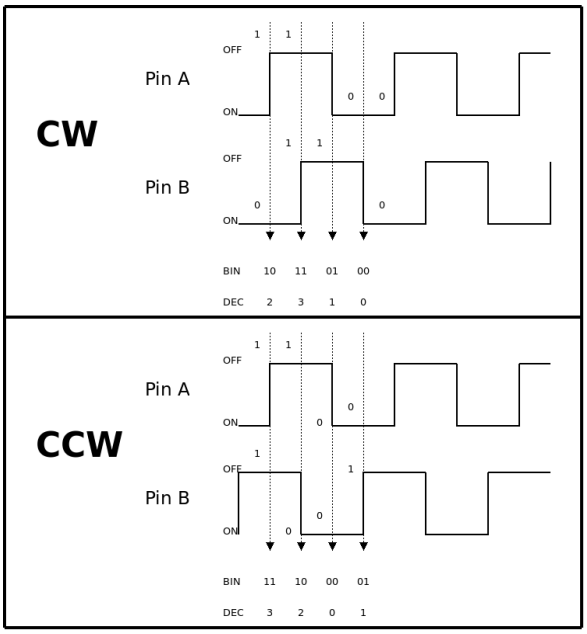


Figure I.7.c Numérisation des signaux de l'encodeur pour déterminer le sens de déplacement de la souris

Gray coding for counter-clockwise rotation		
Phase	A	B
1	0	0
2	0	1
3	1	1
4	1	0

Gray coding for clockwise rotation		
Phase	A	B
1	1	0
2	1	1
3	0	1
4	0	0

Tableau I.1 Le codes de gray pour les deux sens de rotation

I.2.4 Première souris optique

Les souris à boule ont des parties mécaniques mobiles qui ont tendance à ramasser la poussière de la surface de travail et à encrasser les rouleaux capteurs, ce qui exige un nettoyage interne régulier. Les souris optiques ont l'avantage de n'avoir aucune partie mobile dans son mécanisme.

Développée en 1982 par Richard F. Lyon et Steven Kirsch de Mouse System, elle utilise un tapis métallique, rigide et quadrillé (figure I.8.a). Cette souris a deux LED, une LED rouge et une LED infrarouge qui émettent leurs lumières sur le tapis (figure I.8.b). Les lignes horizontales sur le tapis de la souris réfléchissent la lumière rouge, et les lignes verticales réfléchissent la lumière infrarouge (figure I.9). La souris a aussi deux paires de capteurs

optiques (photodiodes). Une paire de photodiodes pour détecter les déplacements suivant x et une autre paire pour détecter les déplacements suivant y (figure I.8.b).

Figure I.8 **a.** Première souris optique avec son tapis métallique - **b.** Première souris optique vue de l'intérieur

On obtient à la sortie de chaque capteur un signal électrique rectangulaire. Les niveaux hauts du signal correspondant à la détection d'une ligne sur le tapis, et un niveau bas à l'absence de ligne. La raison pour laquelle les capteurs de déplacement suivant x et y sont en paire, est de permettre la détection du sens des déplacements suivant ces deux directions. L'inconvénient de ce type de souris est qu'on doit toujours veiller à ce que cette dernière soit bien alignée avec son tapie pour avoir une mesure précise de sa position **[v6]**.

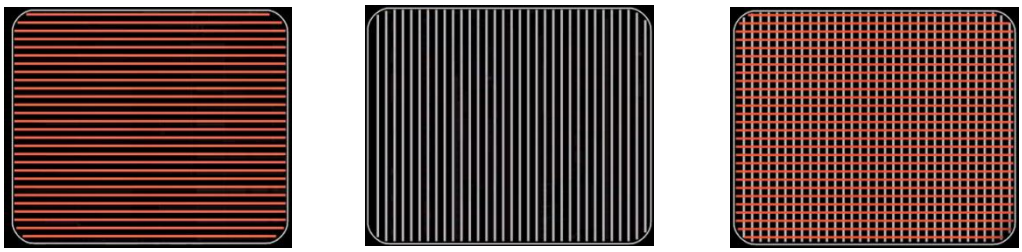


Figure I.9 Les lignes horizontales du tapis réfléchissent la lumière rouge et les lignes verticales réfléchissent la lumière IR

I.2.5 La sourie optique

La souris optique que nous utilisons encore de nos jours a été développée en 1999 par Agilent Technologies et, contrairement au modèle précédent, elle ne nécessite pas de tapis spécial. Elle est constituée principalement d'un capteur (une caméra), une LED et un microcontrôleur (figure I.10).

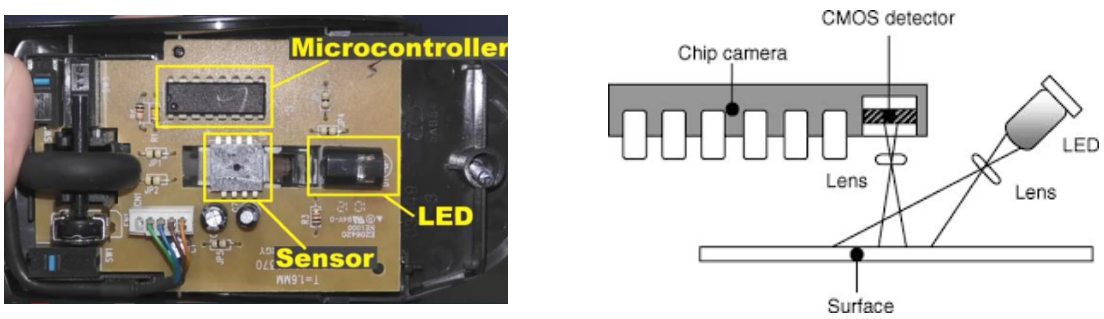


Figure I.10 Les principaux constituants d'une souris optique

Son fonctionnement consiste à capturer avec la caméra des images d'une très petite portion carré de la surface de travail (carré de quelques millimètres de coté). L'image obtenue est donc un grossissement de cette surface puisqu'elle montre les détails de sa texture (figure I.11). Les irrégularités de la surface de travail, qui ne sont pas apparents à l'œil nu, sont visibles dans l'image à cette échelle de grossissement. Et pour mieux faire apparaitre ces irrégularités dans l'image, la surface est éclairée avec la LED de façon incliné pour obtenir les ombres des irrégularités **[v1] [v2] [v3] [v6]** (figure I.11).

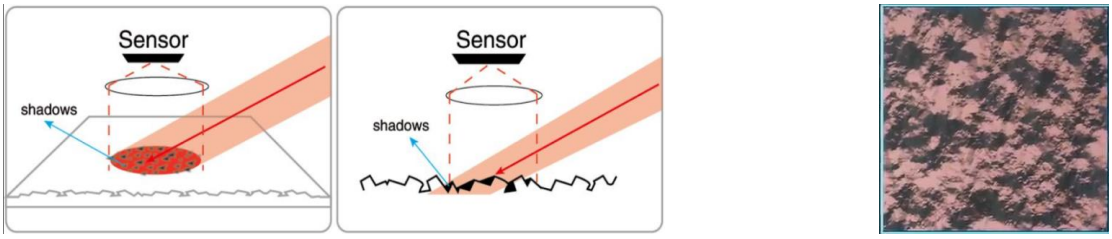


Figure I.11 Capture d'une image des irrégularités de la surface

La caméra capture plus de 1000 images par seconde. Pour déterminer les mouvements de la souris sur la surface de travail, ces images sont traitées en comparant les images successives. Comme l'étude des souris optiques fait l'objet de ce mémoire, on se contentera ici d'en donner seulement le principe de fonctionnement et, on l'étudira plus en détails dans les paragraphes suivants.

I.2.5.1 Molette (Scroll-wheal)

Les souris récentes sont toutes équipées d'une molette qui sert à déplacer la glissière de la bare de défilement vertical dans un lecteur de documents (htm, pdf, word, …). Cette molette sert à parcourir des documents avec plusieurs pages ou avec une longue page unique.

Le capteur de molette se base sur un principe mécanique simple. Placé sur l'axe de la molette, il ouvre et ferme deux interrupteurs de façon continue quand la molette est tournée. Les commutations de ces deux interrupteurs sont déphasées pour permettre de déterminer le sens de rotation de la molette. En reliant ce capteur a une résistance de tirage et une tension de 5V on obtient à sa sortie deux signaux électiques réctangulaires déphasés, grace auxquels on peut

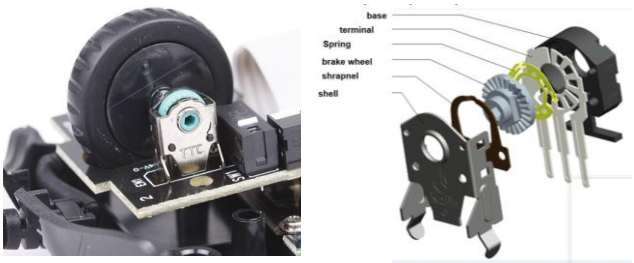


Figure I.12 Molette de souris et les constituants de son capteur

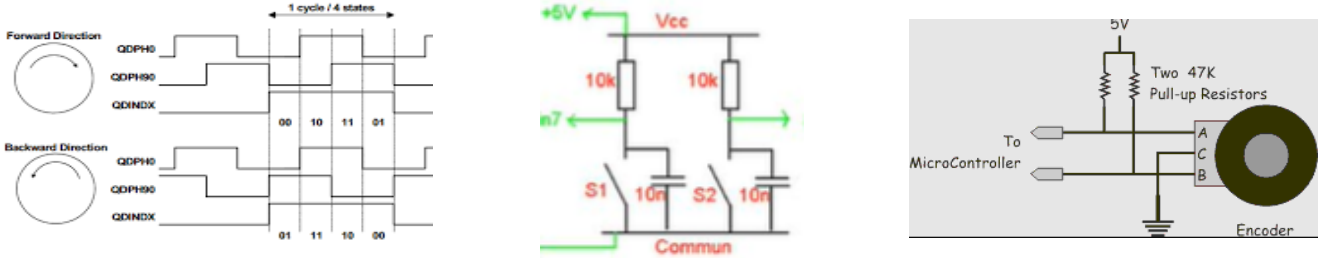


Figure I.13 **a.** Signaux de sortie d'un capteur de molette – **b**. Circuit équivalent d'un capteur de molette – **c**. Brochage d'un capteur de molette

déterminer la vitesse et le sens de parcours d'un document en tournant la molette.

I.2.6 Pavé et écran tactile

C'est une surface sensible au toucher utilisée comme pointeur sur les ordinateurs. Il peut être séparé ou bien intégré au clavier, comme sur de nombreux portables. Son avantage est de pouvoir l'utiliser tout en conservant les mains proches du clavier. Il existe plusieurs types de capteurs tactiles (résistif, capacitif, à infrarouge, …), mais le plus courant est le capteur tactile capacitif.

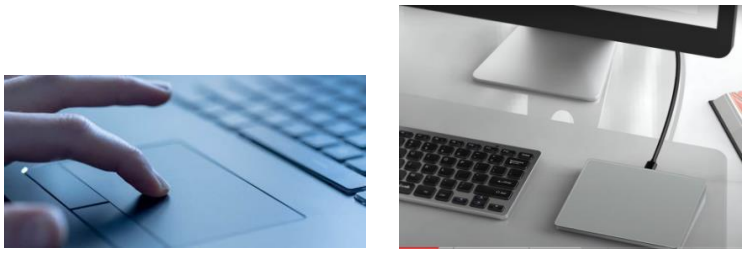


Figure I.14 Pavé tactile intégré et pavé tactile externe

De façon générale, les capteurs capacitifs sont constitués de plusieurs couches (figure I.15). Les couches interne et externe sont des conducteurs d'électricité, alors que les couches intermédiaires sont des isolants. Si bien que le pavé ou l'écran tactile se comporte comme un condensateur. Quand on ne touche pas le capteur tactile, le champ électrique entre ses deux couches conductrices est uniforme. L'approche du doigt sur le capteur tactile crée une concentration de charges électriques sous le doigt. La position de cette concentration est déterminée grâce à la grille conductrice et un traitement informatique intégré.

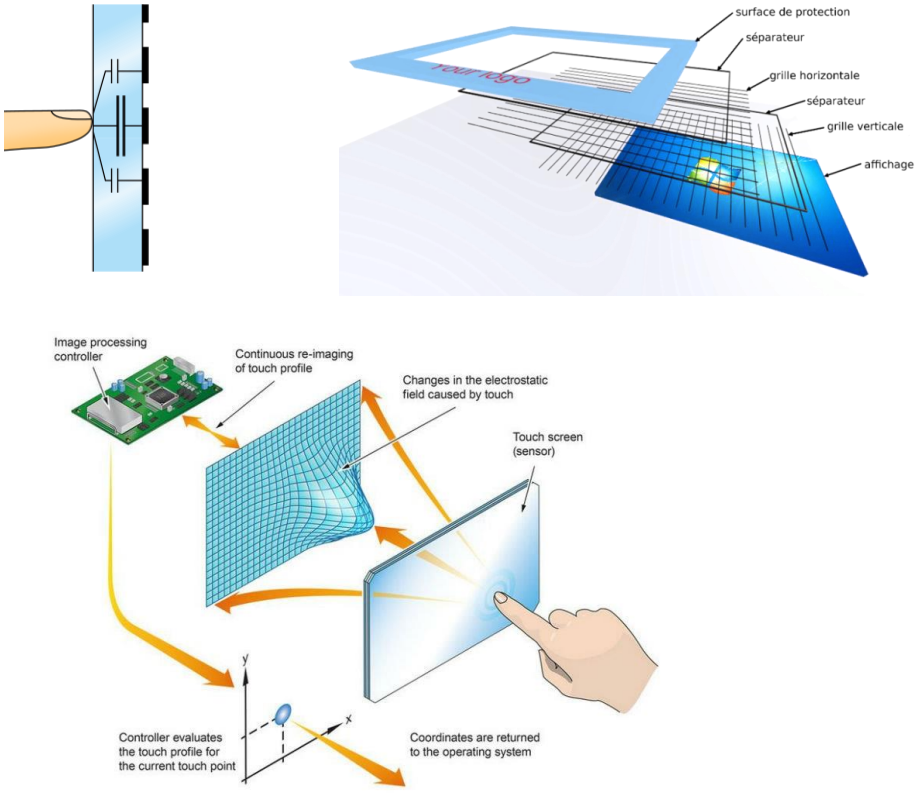


Figure I.15 Constituants et principe de fonctionnement d'un écran/pavé tactile capacitif

I.2.7 Souris spéciales

Depuis l'invention de la première souris d'ordinateur jusqu'à ce jour les concepteurs n'ont pas cessé d'améliorer les souris des points de vu performance et ergonomie. La souris étant un outil de travail à usage intensif une mauvaise ergonomie peut provoquer des problèmes de santé, comme une tendinite. D'où la conception des « **souries verticales** » qui offrent un plus grand confort à l'usager (figure I.16).

Figure I.16 Souris verticales

Il existe aussi d'autres formes de souris comme la « **souris stylet** » qui se tient à la main comme un stylo (figure I.18), et la « **souris track-point** » en forme de bouton intégrée dans un clavier (figure I.17).

Figure I.17 Souris track-point

Figure I.18 Souris stylet

La « **souris à doigt** » avec trackball quant à elle permet un usage de la souris sans plan de travail. Elle est généralement sans fil.

Figure I.19 Souris à doigt

La « souris 3D » très pratique pour les concepteur d'objets 3D et leurs visualisation dans un environnement 3D (architecture, définition de pièces mécaniques).

Figure I.20 Souris 3D et leurs utilisations

I.3 Souris optique

On a présenté au paragraphe I.2.5 le principe de fonctionnement d'une souris optique de façon générale. Dans ce paragraphe on va étudier plus en détail l'architecture et le fonctionnement de cet appareil de pointage.

I.3.1 Les constituants d'une souris optique

Puisqu'on ne s'intéresse dans ce travail qu'à l'estimation du mouvement de la souris, on ne parlera ici que des constituants qui interviennent dans ce processus et on n'abordera pas les parties en rapport avec les fonctions de boutons droit, centre et gauche, et molette.

On a parlé dans le paragraphe §I.2.5 des trois principaux constituants de la souris qui sont :

- Le **capteur** de souris optique intégrant une caméra qui capture des images d'une très petite portion de la surface de travail (carré de quelques mm de côté), située juste au-dessous du capteur d'image,
- La **LED** qui sert à éclairer cette surface,
- Le **microcontrôleur** qui sert à la communication avec l'ordinateur auquel la souris sera reliée, et à la gestion du capteur, la LED, les boutons droit-centre-gauche, et la molette (figure I.22).

On ajoute à ces constituants la lentille **collimatrice** qui a pour rôle de véhiculer la lumière de la LED vers la surface de travail à capturer de façon à avoir un angle d'incidence bien précis (figure I.21). On rappelle que cette lumière inclinée permet d'avoir une trace des irrégularités

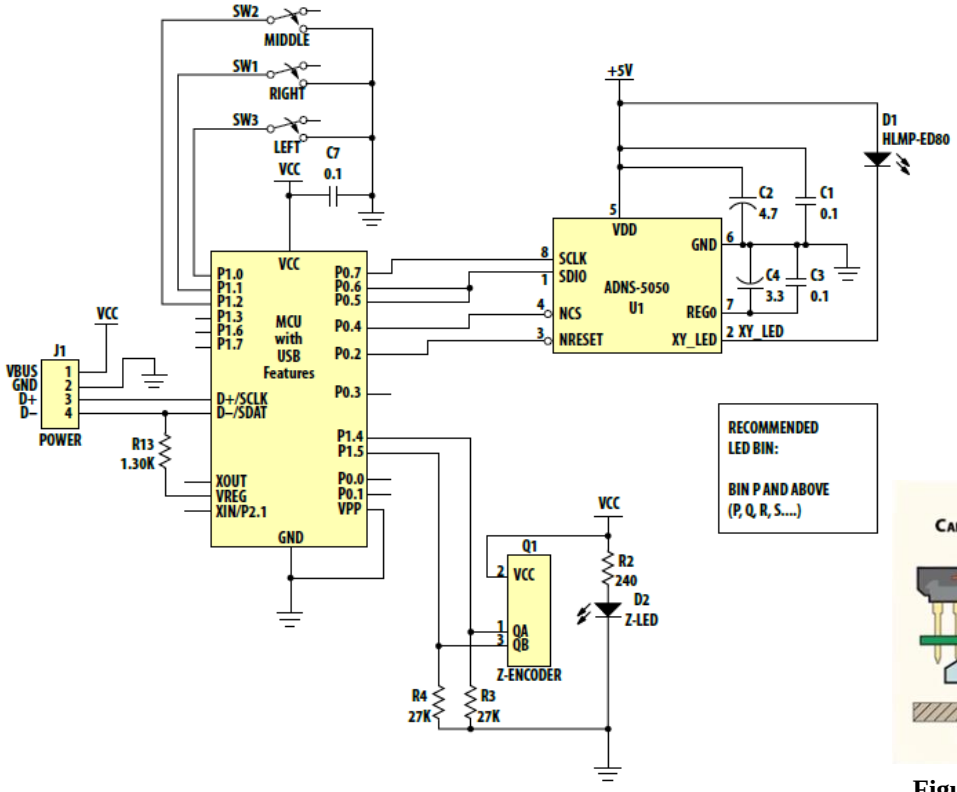


Figure I.22 Partie électronique d'une souris optique

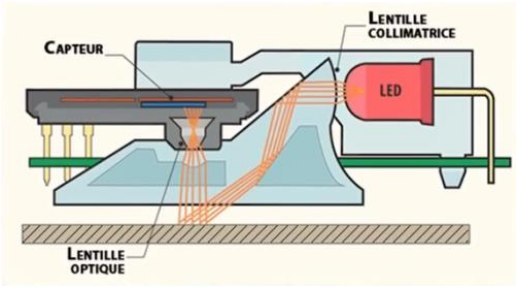


Figure I.21 Partie optique d'une souris d'ordinateur

de la surface de travail sur l'image en créant par cet éclairage, des ombres de ces irrégularités.

I.3.2 Capteur de souris optique

Ce capteur se présente sous forme d'un circuit intégré et ne se limite pas seulement à un simple capteur d'image. En effet, si on ôte le couvre inférieur du boîtier, on peut voir la puce électronique sur laquelle on distingue, dans une petite zone carrée située juste en face de l'ouverture dans le boîtier, la matrice de photosites du capteur d'image (figure I.23).

La (figure I.24) montre un schéma fonctionnel du capteur de souris optique ADNS-5050. On peut aussi distinguer sur l'image d'une telle puce vue au microscope (figure I.24), les circuits du **DSP**, du **contrôleur de LED**, et de **l'alimentation**. Les images capturées sont localement traitées par le DSP pour déduire le déplacement de la souris. Le schéma de la (figure I.25) issu du brevet d'invention du premier capteur de souris optique montre d'autres détails du capteur, comme les composants du système d'acquisition (amplificateur, ADC, étage multiplexeur), et quelques modules du DSP (modules de filtrage, de corrélation et d'interpolation).

En résumé, le système d'acquisition d'images capture des images de la surface éclairée par une LED. Ces images sont traitées par le **DSP** pour déterminer les déplacements relatifs Δx et Δy. Un microcontrôleur externe lit ces informations (Δx et Δy) à partir du **port série** du capteur. Le microcontrôleur transmet ces données à l'ordinateur à travers le port USB ou PS2.

Pour économiser la consommation d'énergie électrique de la souris la LED ne fonctionne en mode DC que lorsqu'un mouvement est détecté. À l'état de repos la LED clignote. D'où le rôle du bloc « **LED DRIVE** » (**contrôleur de LED**). C'est pour cela qu'on voit la LED augmenter pendant le déplacement la souris.

Le port série est utilisé pour régler les paramètres du capteur, et lire l'information de mouvement en accèdent aux **registres** internes du capteur.

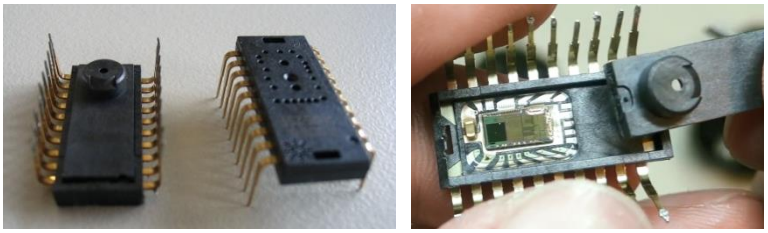


Figure I.23 Capteur de souris optique vu de l'extérieur et de l'intérieur

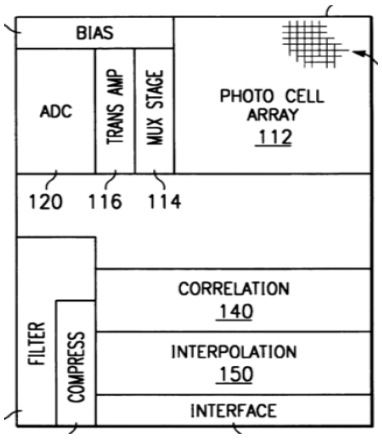


Figure I.25 Schéma fonctionnel d'un capteur de souris optique plus détaillé **[d1]**

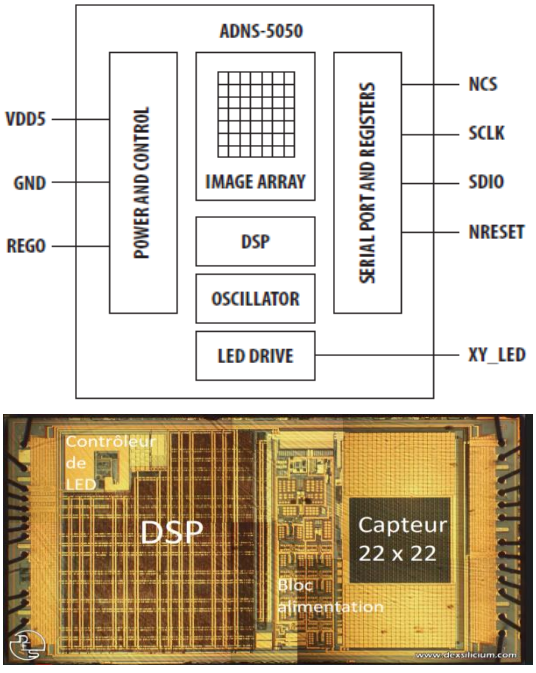


Figure I.24 Schéma fonctionnel d'un capteur de souris optique (**en haut**) **[d4]** – Circuit intégré d'un capteur optique vu au microscope (**en bas**) **[V4]**

I.3.3 Évaluation des mouvements d'une Souris optique

Sur la (figure I.26), l'image à gauche représente la 1ière image de la surface de travail capturée par la souris, et l'image à droite représente la 2ième image capturée après un déplacement de la souris. Ces images ont une dimension 25x25. On peut comparer entre ces deux images pour chercher les zones semblables. Pour cela, on considère un bloc de 7x7 pixels dans la première image, qu'on va considérer comme bloc de référence, et on recherche dans la deuxième image le bloc qui ressemble le plus au bloc de référence. Comme la souris s'est déplacée depuis la première capture, le bloc recherché ne va pas se trouver au même emplacement qu'au paravent.

Une fois le boc le plus ressemblant trouvé, on détermine la distance entre sa position dans l'image actuelle et sa position sur la première image. Remarquons que le déplacement de la souris est dans le sens inverse du déplacement du bloc trouvé.

Pour chercher l'emplacement de ce carré de pixels dans la deuxième image on calcule la **corrélation** (équation.1) entre ce carré et chaque carré de pixels de même taille (7x7) de la deuxième image. Il suffit pour cela de glisser le carré sur la deuxième image de façon à la parcourir de bout-en-bout (figure I.29). La corrélation permet une mesure de la ressemblance entre deux portions d'images. Il existe plusieurs formules de corrélation. Dans notre cas, on a utilisé la corrélation « **Différence au carré** » (**Squared difference correlation**).

$$R_{sq\_diff} = \sum_{x',y'} [T(x',y') - I(x+x',y+y')]^2 \qquad (1)$$

T : Bloc d'image de référence
I : L'image courante
x, y : L'emplacement du bloc de l'image courante
$\mathbf{R_{sq\_diff}}$: Corrélation fonction de x et y

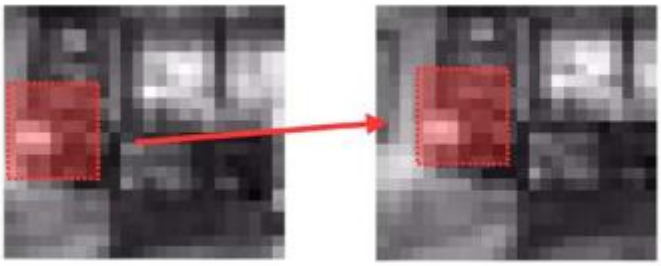


Figure I.26 Comparaison de deux images successives d'une souris optique. Les Carrés de pixels en rouge sont semblables

Pour déterminer le déplacement, on calcule la distance entre la position du bloc de référence et la position du bloc le plus ressemblant dans la deuxième image.

Figure I.27 Détection des déplacements de la souris à partir des déplacements d'un carré de pixels d'une image à l'image qui succède

En vision artificielle, le processus qu'on a décrit est appelé « **Block-Matching »** dans lequel on recherche dans l'image **courante** un bloc similaire à un bloc de l'image **précedente**, on détermine ainsi le vecteur de mouvement qui relie ces deux blocs. Cette technique est utilisée dans beacoup d'autres domaines comme : la reconnaissance et le suivi d'objet, la mise en correspondance des images stéréo, et l'estimation de mouvement pour la compression vidéo.

La fréquence d'acquisition des images du capteur dépend de la vitesse max de déplacement de la souris puisqu'on ne peut pas déterminer le déplacement s'il n'y a pas de pixels communs entre les deux images (figure I.28). On choisi généralement comme bloc de pixels de référence le bloc situé au milieu de l'image (figure I.29).

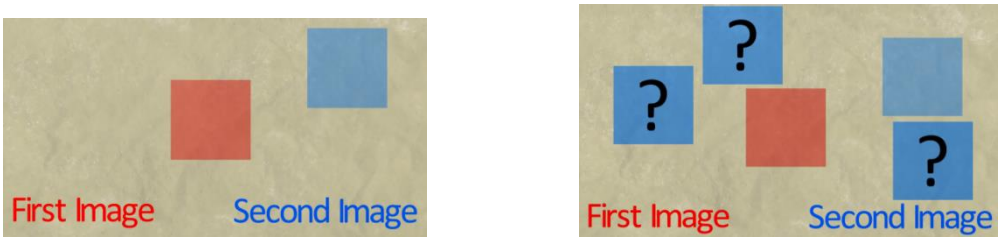


Figure I.28 Le déplacement de la souris ne peut être détecté si les parcelles dans les deux images successives ne se chevauchent pas

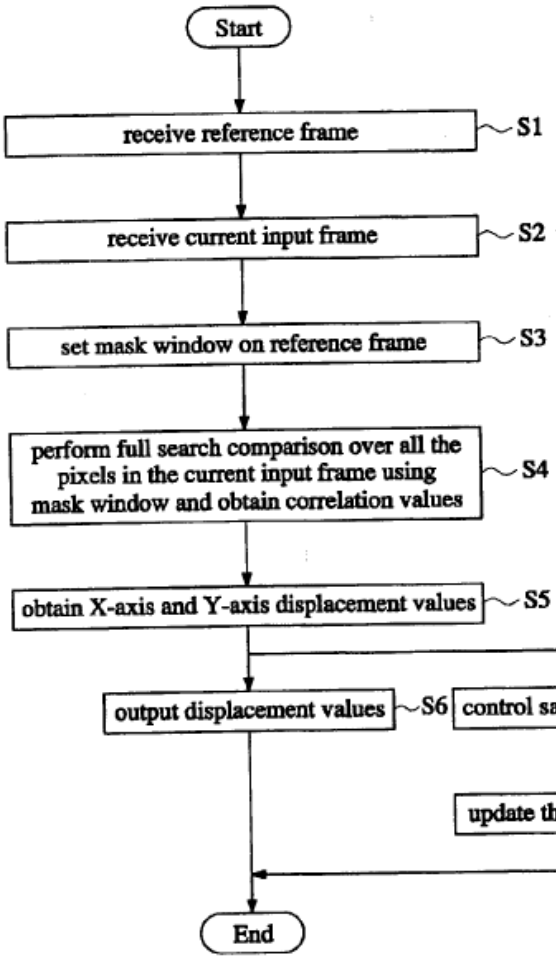


Figure I.30 Organigramme des traitements d'image pour la détection des déplacements d'une souris optique **[d2]**

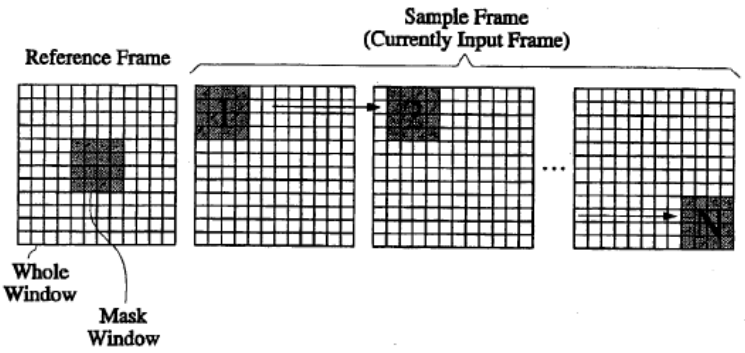


Figure I.29 Image avec un bloc de pixels de référence choisi au centre de l'image (**à gauche**) - On compare ce bloc avec les différents blocs de l'image suivante pour trouver le plus ressemblant (**à droite**) **[d2]**

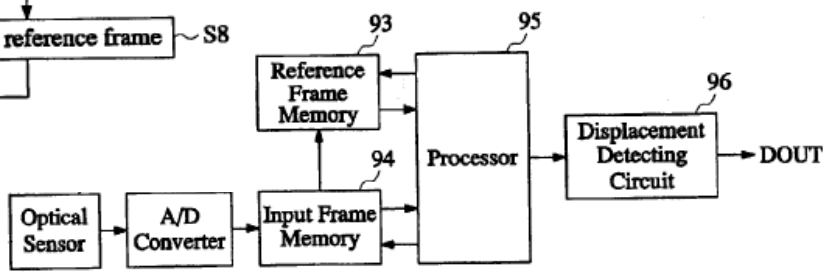


Figure I.31 Système de traitement pour la détection des déplacements d'une souris

I.3.4 Souris laser

Les souris optiques à LED ne fonctionnement pas quand elles sont posées directement sur des surfaces en verre ou des surfaces trop brillantes et sans texture ou trop sombres, à moins qu'on utilise un tapis de souris pour que la souris détecte le mouvement. Une souris laser n'a pas cette limite, et peut fonctionner sur ces types de surfaces sans tapis.

La souris laser se base sur le même principe que la souris optique ordinaire. La principale différence entre ces deux types de souris est leur source d'éclairage. Une souris optique ordinaire utilise une lumière de LED rouge ou infrarouge alors qu'une souris laser utilise une lumière laser provenant d'une diode laser.

Comme on peut le voir sur la (figure I.32), une lumière quelconque émise sur une surface très lisse (brillante) est réfléchit suivant un angle bien défini, contrairement à une surface moins lisse qui réfléchit sa lumière presque dans toutes les directions à cause de la rugosité de la surface. Une diode laser on a faisceau de lumière plus étroit que celui d'une LED. Avec une surface très lisse, la lumière laser réfléchie est elle aussi étroite. C'est pour cela que dans une souris laser, le capteur est placé dans la direction de réflexion de la lumière (figures I.33-I.34).

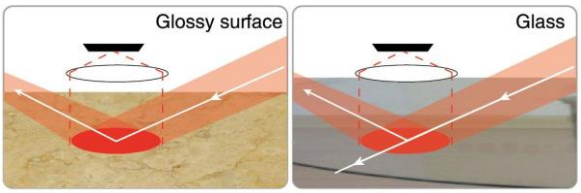


Figure I.32 Réflexion suivant un angle précis dans le cas des surfaces très lisses

En comparant les images d'une surface capturées avec une souris laser et une souris à LED (figure I.35), on peut voir que l'image d'une souris laser contient plus de détails que celle d'une souris à LED. C'est ce qui justifie la précision des souris laser par rapport aux souris à LED. Cependant, les souris laser sont plus chères que les souris à LED parce qu'elles utilisent un émetteur laser et un capteur très précis.

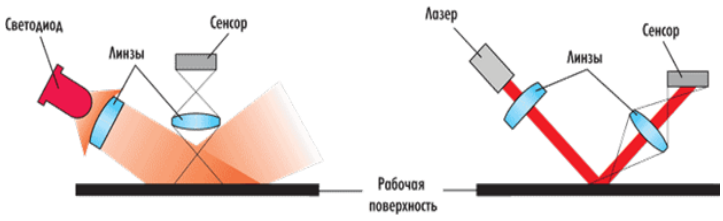


Figure I.33 Configurations de la lentille et du capteur dans les cas d'une souris à LED et une souris Laser

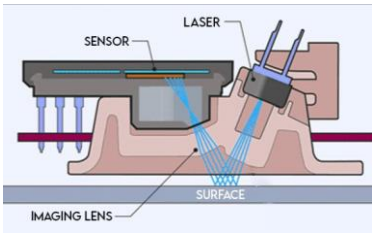


Figure I.34 Système optique d'une souris Laser

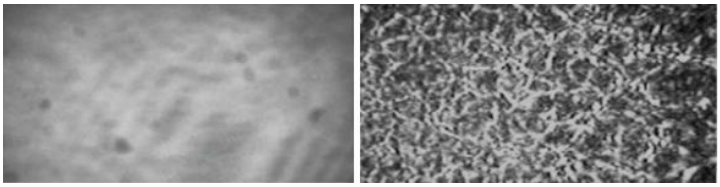


Figure I.35 Image d'une surface lisse capturée avec une souris à LED (**à gauche**) - Image de la même surface capturée avec une souris Laser (**à droite**)

I.4 Capteur ADNS5050

C'est le capteur de souris optique que nous avons utilisé dans notre projet.

I.4.1 Le constructeur

Avago (ex **Agilent**) est le constructeur de ce type de capteur.

- **Agilent** a été formée par la scission de **Hewlett-Packard** (**HP**) en **1999**.
- En **2005**, **Agilent** vend son activité qui produisait des circuits intégrés à semi-conducteurs et devient la société **Avago Technologies**.
- **Avago** annonce en **2006** avoir atteint **600 millions** de capteurs de souris optiques vendus depuis le lancement de son premier capteur en 1999.
- En **2009 Avago** est rachetée par **Broadcom Corporation,** et c'est la date à laquelle elle a cessé de produire des capteurs de souris optiques.

Figure I.36 Logos d'Avago Technologies et Agilent Technolologies

I.4.2 Pourquoi avoir choisi ce type de capteur ?

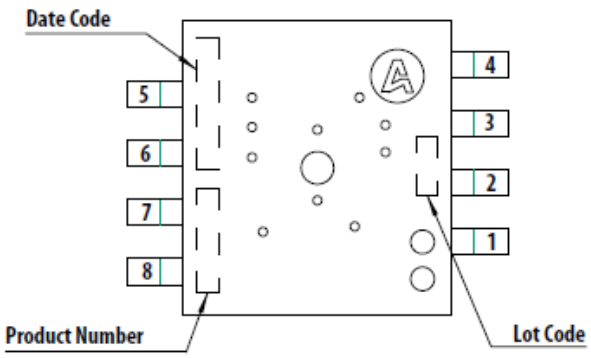
Les capteurs des souris optiques d'Avago (ex Agilent) sont les seuls dont les datasheets expliquent la manipulation de leurs registres internes. C'est ce qui a permis de les exploiter et a inspiré beaucoup d'électroniciens amateurs à les utiliser comme des capteurs de mouvements dans leurs projets. C'est pour cela qu'il aurait été impossible de réaliser la présente application sans une souris Agilent ou Avago. On remarque que puisque ces capteurs ne sont plus produits depuis 2009, il est un peu difficile d'en trouver.

I.4.3 Caractéristiques

- Boîtier DIP 8 pins
- Détection de mouvement très rapides à 30 ips (inch per seconde)
- Auto-ajustement de la fréquence des images (frame rate) pour une performance optimale.
- Résolution par défaut de 500 dpi (dots per inch), ajustable entre 125 et 1375 dpi par pas de 125. Un nombre élevé de DPI permet une précision accrue lors du déplacement du pointeur, pour un usage bureautique, une précision d'environ 800 à 1 600 DPI suffit.
- Interface série three-wire
- Capteur d'images 19x19

I.4.3 Brochage

Les broches **SCLK** (ligne d'horloge), **SDIO** (ligne de donnes d'entrées sortie) et **NCS** (Chip select) sont broches du port série three wire.



Pinout of ADNS-5050 Optical Mouse Sensor

Pin	Name	Description	I/O type
1	SDIO	Serial Port Data Input and Output	I/O
2	XY_LED	LED Control	O
3	NRESET	Reset Pin (active low input)	I
4	NCS	Chip Select (active low input)	I
5	V_{DD5}	Supply Voltage	Power
6	GND	Ground	Ground
7	REGO	Regulator Output	O
8	SCLK	Serial Clock Input	I

Figure I.37 Brochage du capteur ADNS-5050

I.4.5 Les registres internes [d4]

Dans le (Tableau I.2) on a surlgné dans la liste des registres de l'ADNS-5050, les registres qu'on a utilisé dans nos programmes. Tous les registres du capteur sont des registres 8 bits.

Address	Register	Read/Write	Default Value
0x00	Product_ID	R	0x12
0x01	Revision_ID	R	0x01
0x02	Motion	R	0x00
0x03	Delta_X	R	Any
0x04	Delta_Y	R	Any
0x05	SQUAL	R	Any
0x06	Shutter_Upper	R	Any
0x07	Shutter_Lower	R	Any
0x08	Maximum_Pixel	R	Any
0x09	Pixel_Sum	R	Any
0x0a	Minimum_Pixel	R	Any
0x0b	Pixel_Grab	R/W	Any
0x0c	Reserved		
0x0d	Mouse_Control	R/W	0x00
0x0e – 0x18	Reserved		
0x19	Mouse_Control2	R/W	0x08
0x1a – 0x21	Reserved		
0x22	LED_DC_Mode	R/W	0x00
0x23 – 0x39	Reserved		
0x3a	Chip_Reset	W	N/A
0x3b – 0x3d	Reserved		
0x3e	Product ID2	R	0x26
0x3f	Inv_Rev_ID	R	0xfe
0x40 – 0x62	Reserved		
0x63	Motion_Burst	R	0x00

Tableau I.2 Liste des registres internes du capteur ADNS-5050

Dans ce qui suit on va expliquer le rôle de chacun de ces registres, et comment les utiliser.

Registre : **Product_ID** (lecture)
Adresse : 0
Valeur : 0x12
Usage : On utilise généralement ce registre pour vérifier que la communication série est fonctionnel en vérifiant que sa valeur est bien égale à 0x12 quand on lit le registre.

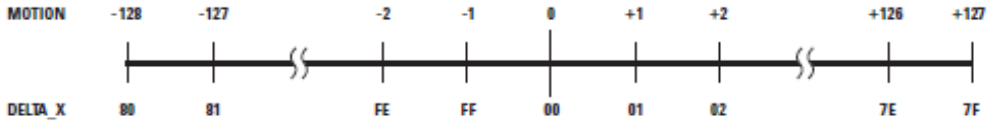
Registre : **Motion** (lecture)
Adresse : 2
Valeur après un reset : 0
Usage : Le bit MOT indique s'il y a eu un mouvement de la souris depuis la dernière lecture du registre. La valeur par défaut du bit MOT est 0.

Mot = 0 ➔ Pas de mouvement de la souris
Mot = 1 ➔ il y a eu mouvement de la souris (Dans ce cas on peut lire les valeurs des registres Delte_X et Delta_Y)

Bit	7	6	5	4	3	2	1	0
Field	MOT	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

---

Registres : **Delta_X** et **Delta_Y** (lecture)
Adresse : 3 et 4
Type de données : Nombre de 8 bits en complément à 2
Usage : Le registre Delta_X donne le mouvement X, et le registre Delta_Y donne le mouvement Y en nombre de points depuis la dernière lecture. La lecture d'un de ces registres l'efface.



Remarque : Delta_X doit être lu avant Delta_Y

Registre : **Pixel_Grab** (lecture/écriture)
Adresse : 0xb
Valeur après un reset : 0
Usage : Ce registre contient la valeur d'un pixel d'une image codé sur 7 bits ($b_6..b_0$). Si le MSB (b_7) du registre est mis à « 1 » la valeur qu'il contient est valide. Quand on lit un pixel, le registre Pixel_Grab est chargé avec la valeur du prochain pixel. Il faut 361 lectures du registre pour lire une image complète. Une écriture sur ce registre va remettre le remettre à zéro et le charger avec la valeur du pixel 0 dans l'image suivante.

Bit	7	6	5	4	3	2	1	0
Field	Valid	PD_6	PD_5	PD_4	PD_3	PD_2	PD_1	PD_0

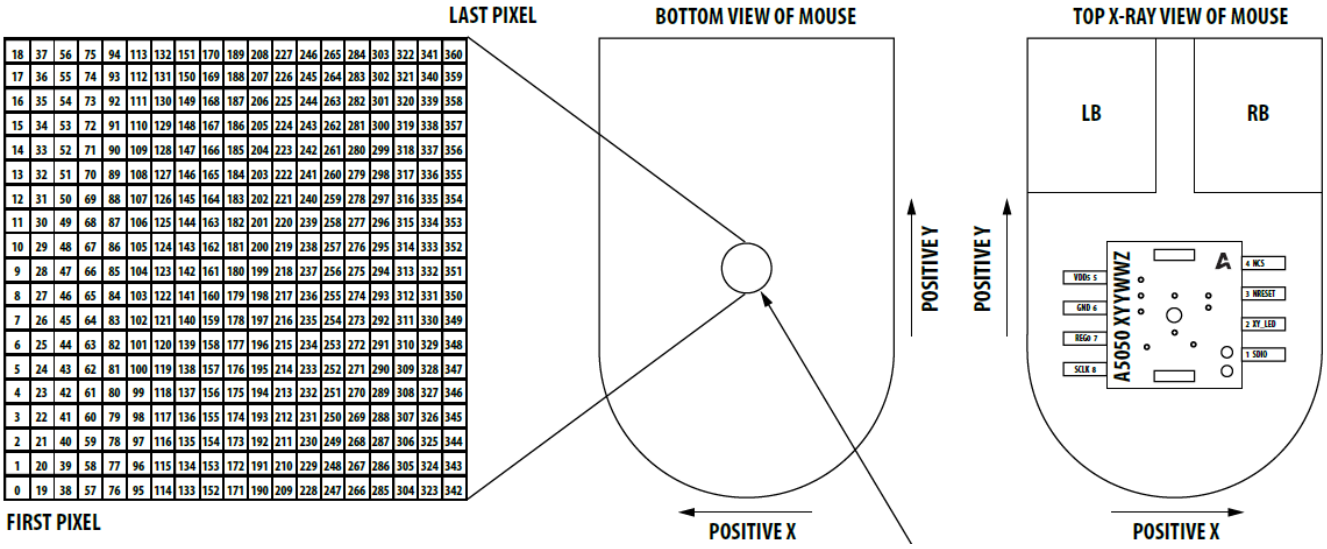


Figure I.38 Numérotation des pixels du capteur d'image et indication des sens de déplacements positif et négatif

---

Registre : **Mouse_Control2** (écriture)
Adresse : 0x19
Valeur après un reset : 8
Usage : Définir la résolution
Remarque : Le bit 0 doit être mis à « 1 » pour qu'une mise à jour de la résolution puisse prendre effet.

Bit	7	6	5	4	3	2	1	0
Field	Reserved	Reserved	Reserved	RES_EN	RES	RES	RES	RES

Field Name	Description
RES_EN	= **0 Disable RES[3:0] setting.** = 1 Enable RES[3:0] setting.
RES [3:0]	= 0b0001: 125 CPI = 0b0010: 250 CPI = 0b0011: 375 CPI = 0b0100: 500 CPI = 0b0101: 625 CPI = 0b0110: 750 CPI = 0b0111: 875 CPI = **0b1000: 1000 CPI** = 0b1001: 1125 CPI = 0b1010: 1250 CPI = 0b1011: 1375 CPI

Registre : **LED_DC_Mode** (lecture/écriture)
Adresse : 0x22
Valeur après un reset : 0
Usage : En écrivant la valeur 0x80 dans ce registre, on force la LED à fonctionner tout le temps en mode DC.

Bit	7	6	5	4	3	2	1	0
Field	LM_7	LM_6	LM_5	LM_4	LM_3	LM_2	LM_1	LM_0

Registre : **Chip_Reset** (écriture)
Adresse : 0x3a
Valeur après un reset : 0
Usage : En écrivant dans ce registre la valeur 0x5a, on réinitialise le capteur

Bit	7	6	5	4	3	2	1	0
Field	CR_7	CR_6	CR_5	CR_4	CR_3	CR_2	CR_1	CR_0

Il y a deux façons de faire un reset du composant, soit en mettant la pin **NRESET** à l'état **bas**, soit en écrivant la valeur **0x5a** dans le registre **Chip_Reset** (adresse 0x3a).

I.4.6 Opérations d'écriture et de lecture

Une opération **d'écriture** est définie comme des données allant du microcontrôleur au circuit ADNS-5050. Une opération de **lecture** est définie comme des données allant du circuit ADNS-5050 au microcontrôleur. Ces données sont véhiculées sur la ligne **SDIO**.

Dans ces deux opérations la donnée est composée de deux octets. Le premier octet est envoyé par le microcontrôleur, il contient l'adresse (sur sept bits) et un MSB dont la valeur indique la direction des données. **MSB=1** dans le cas d'une **écriture**, et **MSB=0** dans le cas d'une **lecture**. Le deuxième octet contient la donnée. Cette donnée est envoyée par le microcontrôleur dans le cas d'une écriture, et elle est envoyée par le circuit ADNS-5050 dans le cas d'une lecture. La figure I.39 montre le chronogramme d'un cycle d'écriture, et la figure I.40 montre le chronogramme d'un cycle de lecture.

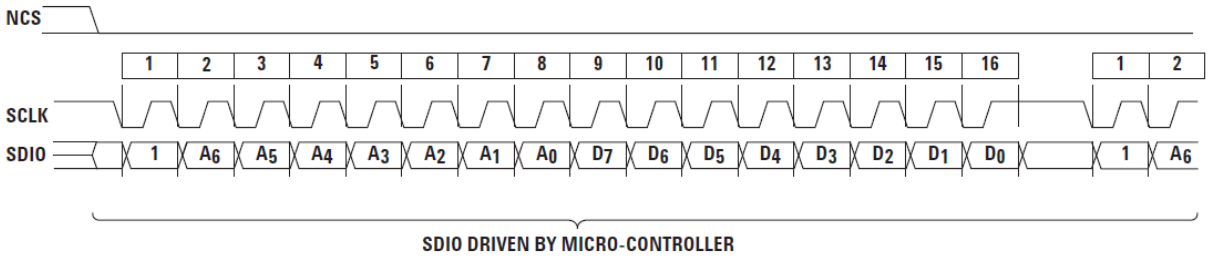


Figure I.39 Chronogramme d'un cycle d'écriture

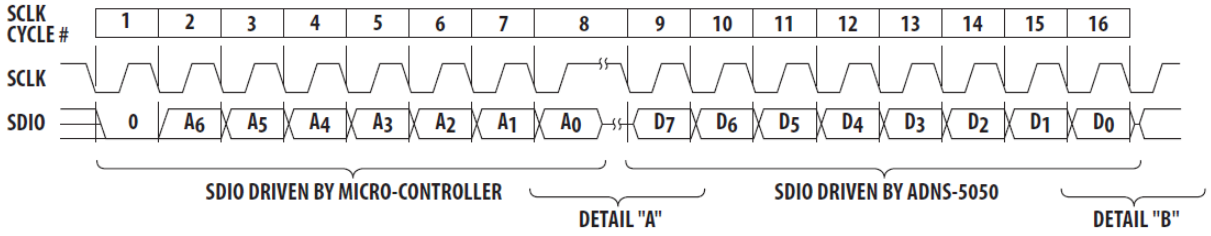


Figure I.40 Chronogramme d'un cycle de lecture

I.5 Conclusion

On a présenté dans ce chapitre la souris d'ordinateur de façon général et son évolution au cours des années passés. Ensuite, on a parlé en particulier de la souris optique : ses constituants, son capteur, les traitements qui détermine ses déplacements. Et enfin, on a donné quelques détails sur le fonctionnement du capteur ADNS-5050 qu'on a utilisé dans notre projet

Chapitre II
Moyens matériels et logiciels utilisés

II.1 Introduction

Dans ce chapitre on va présenter les moyens matériels et logiciels employés pour atteindre notre objectif qui est de déterminer les coordonnées (x,y) des déplacements relatifs d'une souris optique en se basant seulement sur les images capturées par cette souris.

Parmis les moyens matériels impliqués dans ce système, on a :

1. Une souris optique (PLEOMAX MO-200) avec un capteur d'Avago (ex Logitech) ADNS5050,
2. Une carte Arduino UNO
3. Un ordinateur.

Parmis les moyens logiciels impliqués dans ce système, on a :

1. Un programme sur la carte Arduino pour récupérer des informations du capteur ADNS-5050 (images capturées par la souris) en utilisant la bibliothèque Arduino « Adns5050 »
2. Un programme en C++ et bibliothèque OpenCV sur l'ordinateur qui fait les traitements (correlation) des images captutrées par la souris pour en déduire les déplacements de la souris
3. Un programme qui permet les échanges de données entre le programme de l'Arduino et le programme de traitements d'image en C++.

La figure (II.1) montre le schéma fonctionnel du système réalisé avec toutes les parties matériels et logiciels qu'il comporte. Dans la suite de ce chapitre on va expliquer plus en détail le fonctionnement de ces différentes parties.

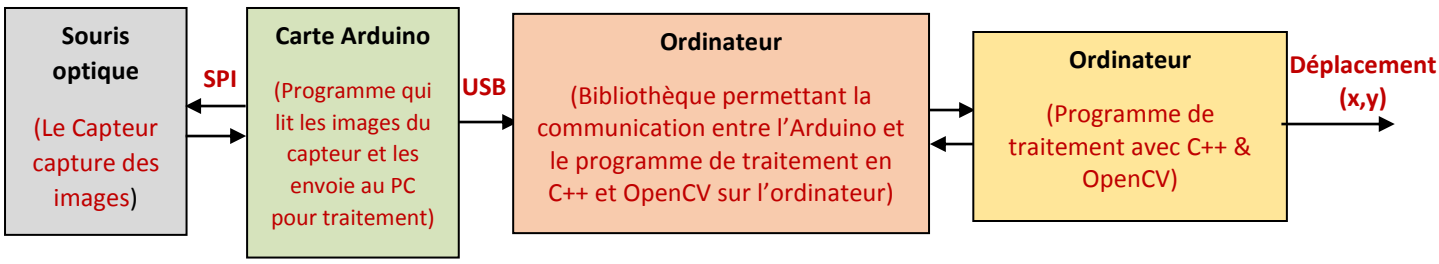


Figure II.1 Schéma fonctionnel du circuit réalisé dans ce projet

II.2 Souris optique utilisée dans le projet

C'est une souris PLEOMAX MO-200 de Samsung (figure II.2) avec un capteur ADNS5050 et contrôleur de souris optique HT82M28A. La figure II.3 montre le schéma de sa carte électronique.

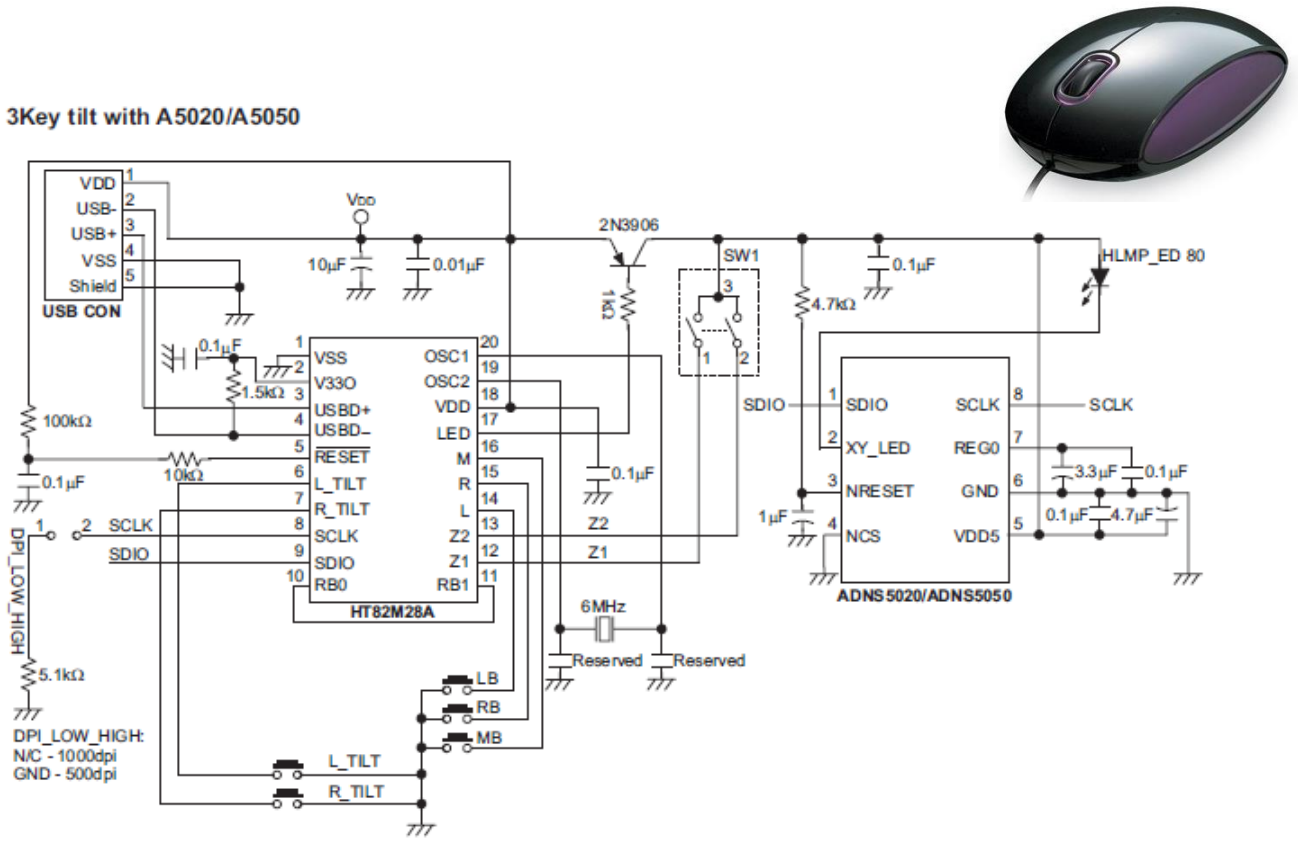


Figure II.2 Circuit de la carte électronique d'une souris PLEOMAX MO-200 (**à gauche**) – Souris PLEOMAX MO-200 (**en haut à droite**) **[d5]**

II.3 Carte de Arduino UNO

II.3.1 Présentation

Arduino est une marque de carte électronique de développement à base de microcontrôleur **ATMEL**, conçue principalement pour un but pédagogique. Elle permet de créer facilement et rapidement des prototypes de systèmes électroniques embarqués, des systèmes de pilotage de robots, des systèmes de domotique (contrôle des appareils domestiques), … etc.

Arduino offre une large gamme de cartes (Méga, Mini, Ethernet… etc.) ayant plus ou moins de ressources (E/S, CAN, mémoires, … etc.) et qui conviennent aux besoins de tous types de projets. La carte « **Arduino UNO** » est à base du microcontrôleur **ATMEGA328**, c'est le type de carte Arduino le plus couramment utilisé.

Pour la programmation des cartes Arduino, un environnement de développement **gratuit « Arduino IDE »** est mis à la disposition des utilisateurs. Cet IDE intègre le compilateur « avr-g++ » pour une programmation en C++, une interface logicielle pour charger les programmes sur le microcontrôleur, ainsi qu'un ensemble de bibliothèques appelé **« Arduino library »**. L'utilisateur peut aussi ajouter des bibliothèques de contributeurs ou écrire ses propres bibliothèques.

Les modules d'origine de l'Arduino sont fabriqués par la société italienne « Smart Projects » et quelques-unes de ces cartes ont été conçues par la société américaine « SparkFun Electronics ».

L' « Arduino IDE » est open source, et les schémas des cartes Arduino (schémas électronique et typons) sont publiés en licence libre (figure II.5). Seul le nom « Arduino » est protégé et ne doit pas être utilisé pour le matériel non authentique. Ce qui a permis l'apparition de marques de compatibles-Arduino, telles que funduino, arduiboy, etc. qui coûtent moins chères que les originaux **[w2]**.

Figure II.4 Logos de l'Arduino et d'**ATMEL**

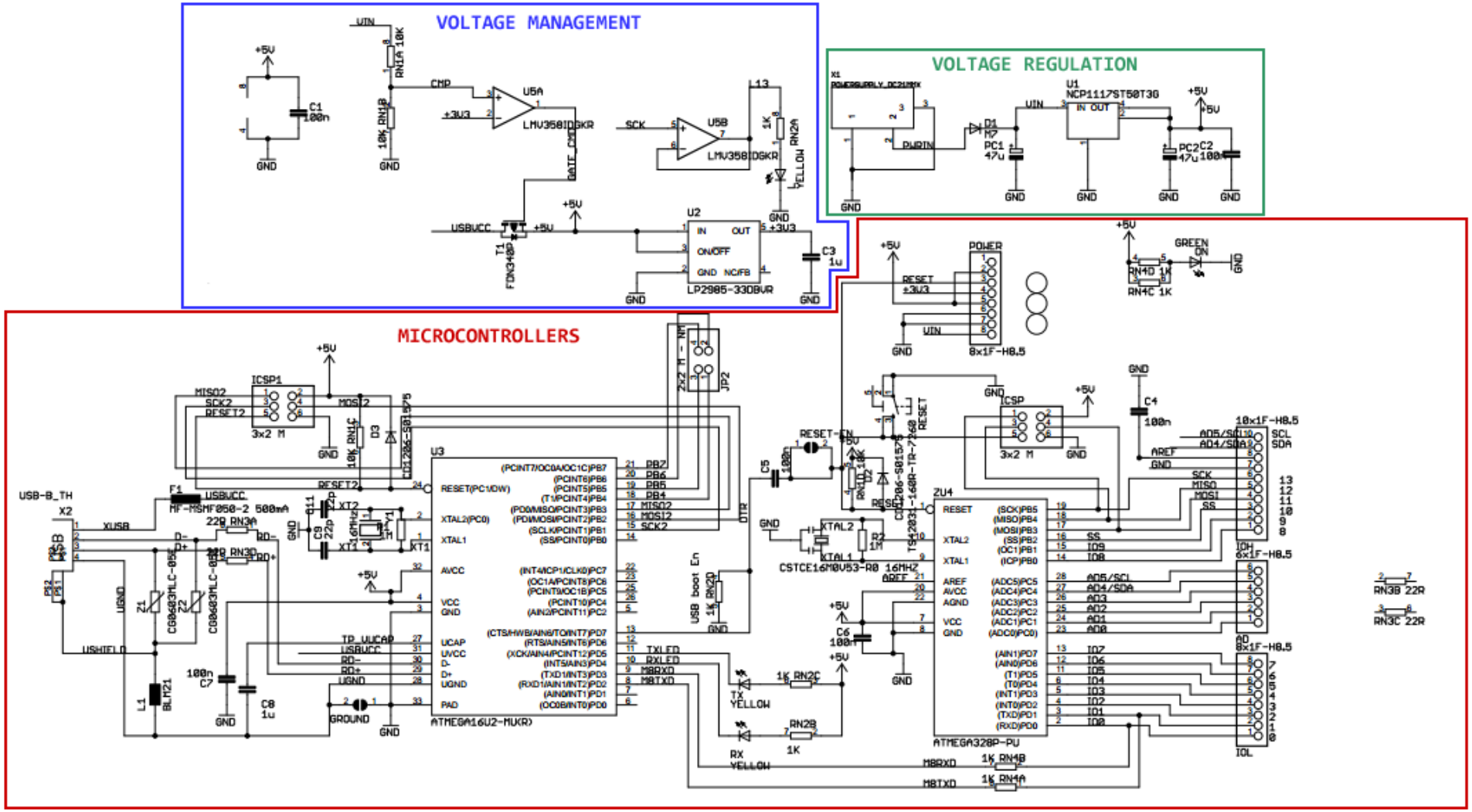


Figure II.5 Circuit électronique d'une carte Arduino UNO

II.3.2 Caractéristiques d'une carte Arduino UNO

Un module Arduino est généralement construit autour d'un microcontrôleur « Atmel » et de composants complémentaires qui facilitent la programmation et l'interfaçage avec d'autres circuits. Chaque module possède au moins un régulateur 5 V et un oscillateur (figure II.8).

Les caractéristiques d'un Arduino UNO sont pour la plus part ceux du microcontrôleur ATmega328, excepté pour la communication USB avec un ordinateur qui n'est pas assurée par le microcontrôleur ATmega328 et se fait à travers le deuxième microcontrôleur (ATmega16u2) présent sur la carte à cet effet (figure II.5).

- Fréquence d'horloge 16Mhz
- Mémoire FLASH à 32Ko : Mémoire de programme.
- Mémoire SRAM (volatile) de 2ko
- Mémoire EEPROM de 1Ko : Mémoire de données
- 14 entrées/sorties numérique
- 6 sorties PWM
- ADC : Résolution 10 bits, 6 entrées analogiques multiplexées
- 3 timers (Deux à 8 bits, et un à 16 bits)
- Communication Série :
    - USART (Universal Serial Asynchronous Receiver Transmitter)
    - SPI (Serial Peripheral Interface)
    - I2C (Inter-Integrated Circuit)

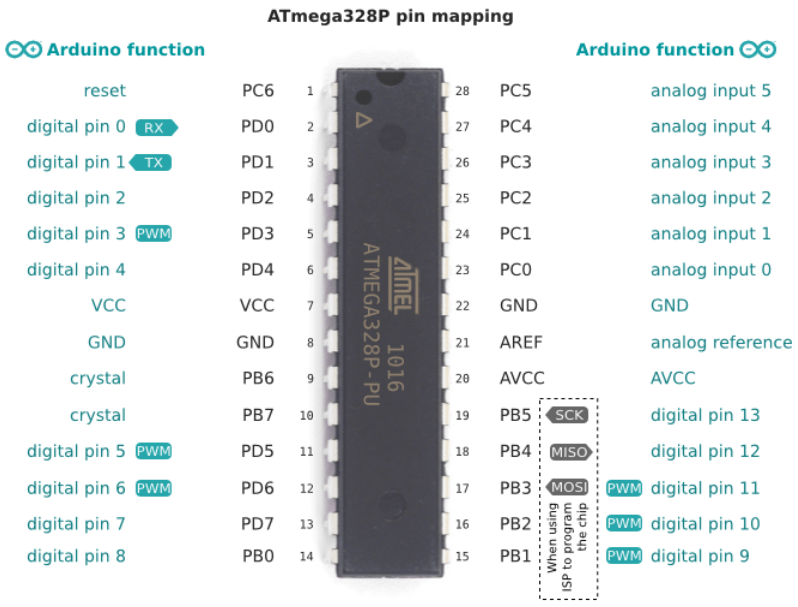


Figure II.6 Brochage du microcontrôleur ATmega328P

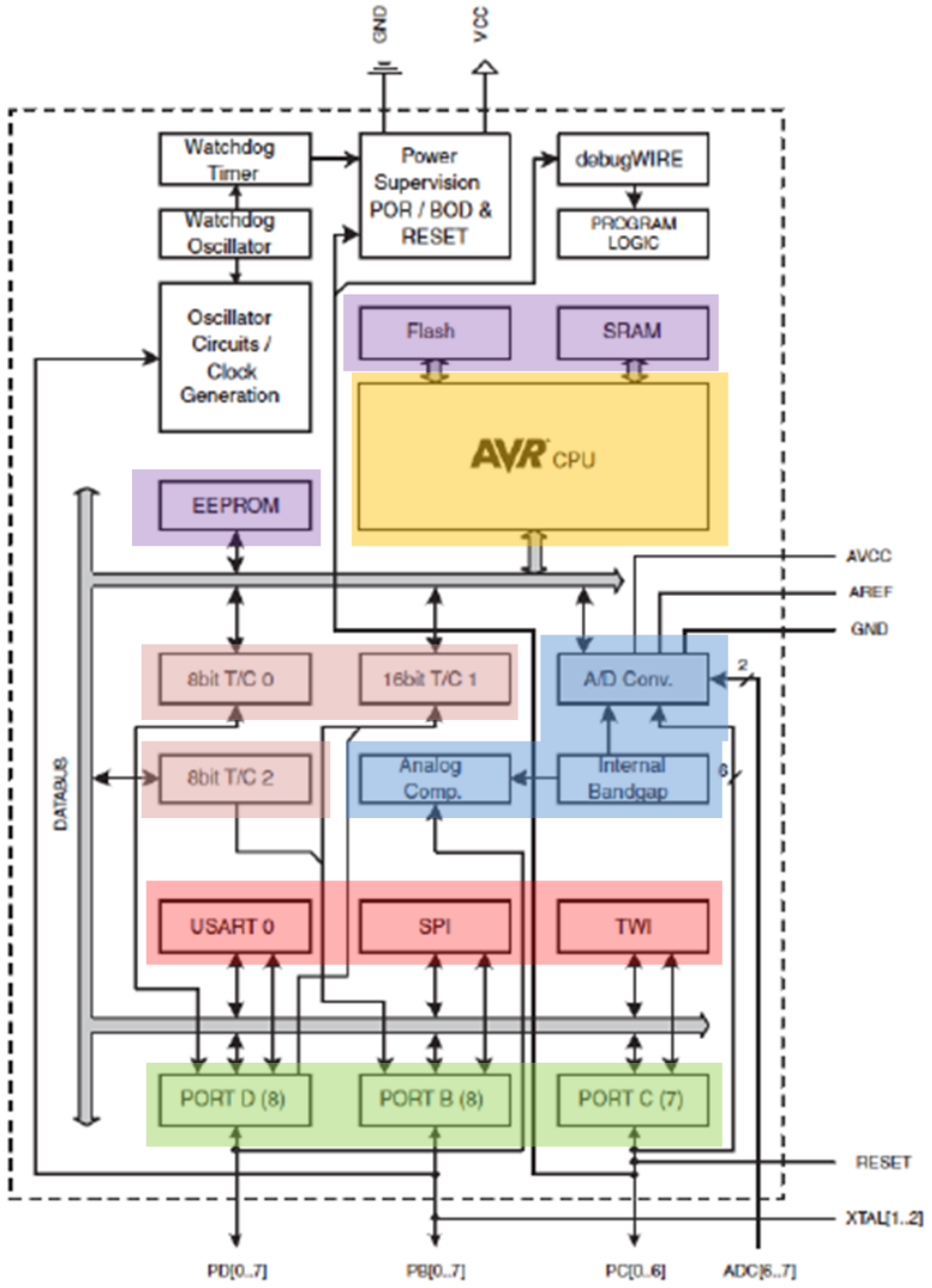


Figure II.7 Schéma fonctionnel d'un microcontrôleur ATmega328

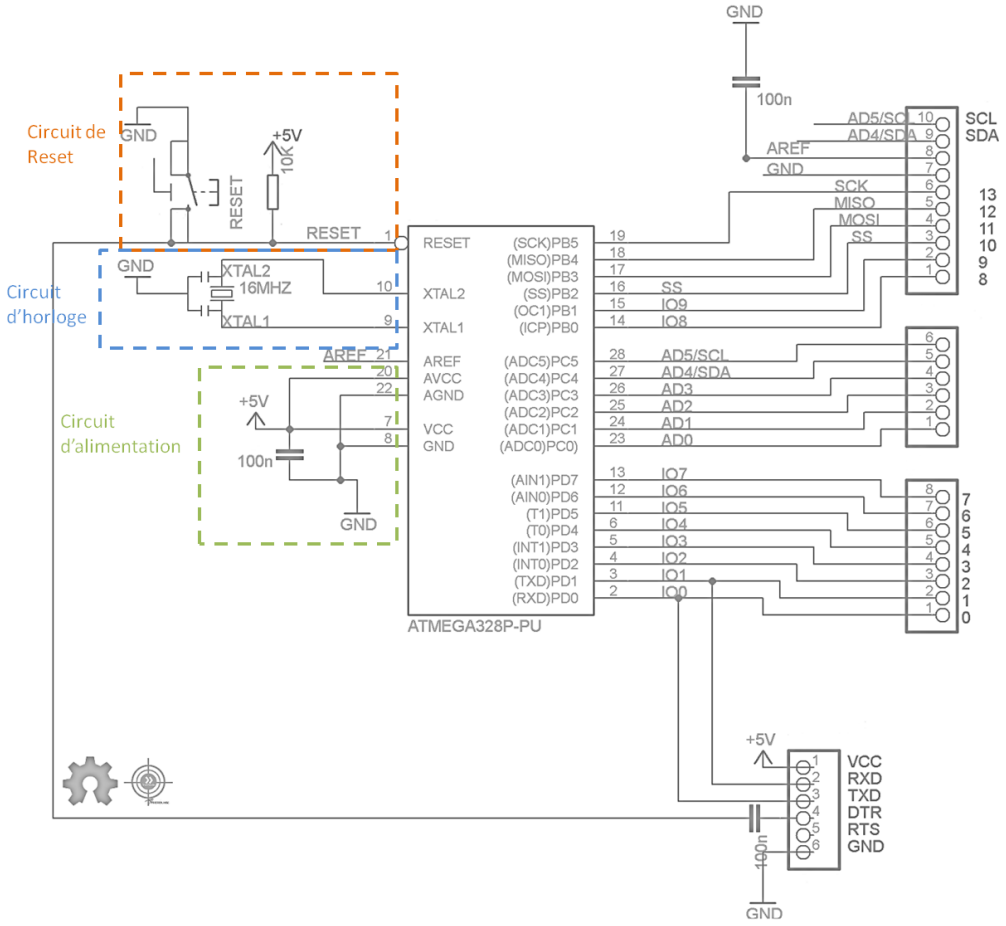


Figure II.8 Schéma électronique de ATmega328P

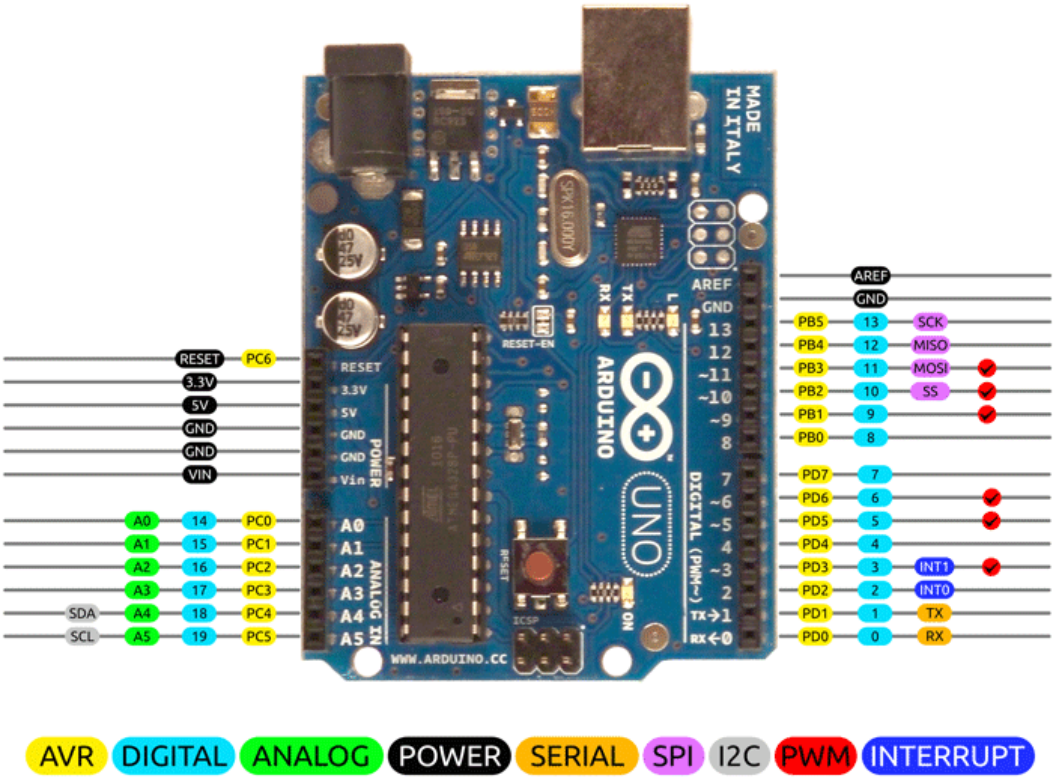


Figure II.9 Brochage de la carte arduino UNO

La carte Arduino est placée entre le capteur de la souris optique et l'ordinateur. Cette carte communique avec le capteur de la souris optique en utilisant une communication SPI, et elle communique avec l'ordinateur en utilisant une communication USART. C'est pour cela que nous allons parlé de ces deux types de communications série et comment on les programme.

II.3.3 Communication USART

USART (Universal Serial Asynchronous Receiver Transmitter) : C'est une interface de communication (transmission et réception de données) série. Comme son nom l'indique, cette interface n'utilise pas de signal d'horloge (asynchrone). Les deux appareils qui veulent communiquer entre eux doivent seulement communiquer à la même vitesse (en bauds = bit par seconde).

Avec une telle communication on peut utiliser au minimum deux lignes : **Tx** (pour la transmission de données) et **Rx** (pour la réception de données). La broche **Tx** du 1ier appareil doit être reliée à la broche **Rx** du 2ième appareil, et la broche **Rx** du 1ier appareil doit être reliée à la broche **Tx** du 2ième appareil (figure II.10).

L'ordinateur est le principal interlocuteur pour une carte Arduino puisqu'on a besoin de transférer des programmes du PC au microcontrôleur; ou réaliser une communication entre les deux. Les ordinateurs actuels ne disposent pas de connecteurs **DB9** et utilisent à la place des connecteurs USB. Le microcontrôleur ATMEGA16U2 présent sur une carte Arduino UNO, supporte la communication USB et a pour tâche de faire les conversions RS232/USB et USB/RS232.

Donc, du coté de la carte Arduino UNO, on écrit un programme d'échange de données comme pour une communication USART, et le microcontrôleur ATMEGA16U2 l'adapte à une communication USB avec l'ordinateur.

Un protocole de communication définit les signaux, les niveaux de tension, les pins utilisés dans la communication, … etc. On donne ci-après quelques règles du protocole de communication :

- Les données échangées sont des octets délimités par un bit « Start » et un bit « Stop ».
- On doit choisir comme vitesse de communication l'une des valeurs suivantes : 9600, 19200, 38400,57600, 115200, 230400, 460800, 1000000, 1500000. On choisit la vitesse en fonction de la longueur du câble entre les deux interlocuteurs. Par exemple, la vitesse 9600 bauds est tolérée pour une longueur de câble inférieur à 15m. Et plus la vitesse de communication augmente, plus la longueur max du câble diminue.
- On peut aussi ajouter des bits de parité paire ou impaire pour contrôler s'il y a des erreurs de transmission (figure II.11).

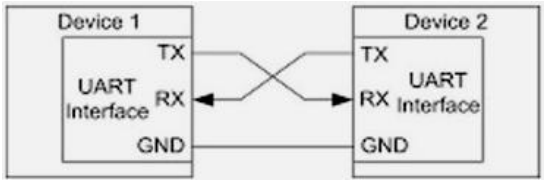


Figure II.10 Port de communication USART

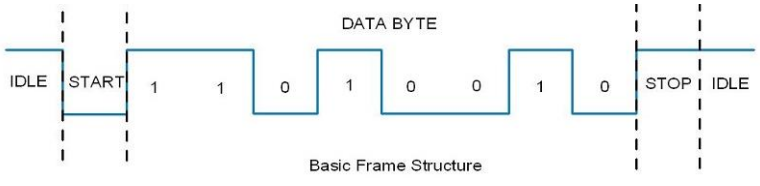


Figure II.11 Structure d'une trame de données dans le protocole de communication USART

- Dans une communication USART de type RS232, le niveau logique « 1 » correspond à une tension entre -3V et -25V, et le niveau logique « 0 » correspond à une tension entre +3V et +25V. Avec une carte Arduino, on utilise une communication USART TTL c.-à-d. 0V pour la valeur logique « 0 », et 5V pour la valeur logique « 1 ».

II.3.3.1 Les règles de la programmation d'une communication USART avec Arduino

Sur l'Arduino UNO, les broches d'une communication USART sont les 0 (Rx) et 1 (Tx) (figure II.9).

`Serial` : Objet port série. Dans la carte Arduino UNO on a seulement un port série. Dans la carte Arduino Mega par exemple, on à 4 ports série à qui on associe les objets Serial, Serial1, Serial2 et Serial3.

Voici quelques fonctions associées à l'objet **`Serial`** :

- **`begin`**`(vitesse)` : Permet d'initialiser la communication
- **`print`**`(donnée)` : Permet à l'Arduino d'envoyer des données
- **`available`**`()` : Cette fonction n'a pas de paramètres. Elle renvoie le nombre d'octets reçus et présents dans le buffer de réception en attente d'être lus.
- **`read`**`()` : Cette fonction n'a pas de paramètres. Elle renvoie le premier caractère arrivé.
- **`parseInt()`** : Cette fonction n'a pas de paramètres. Elle retourne un entier valide lu dans le buffer de réception après un délai configurable (par défaut égal à 1s). Le délai permet d'attendre que l'utilisateur entre au clavier tous les chiffres qui composent l'entier qu'on veut envoyer puisque l'entier n'est pas envoyé d'un seul coups mais chiffre par chiffre (octet par octet). La fonction **`Serial.setTimeout()`** permet de reconfigurer ce délai.

On trouve dans l'IDE Arduino l'outil « **Moniteur série** » qui permet d'afficher les données envoyées par une carte Arduino à l'ordinateur, ainsi que d'envoyer des données à la carte Arduino à partir de l'ordinateur.

II.3.4 Communication SPI

- Une liaison **SPI** (Serial Peripheral Interface) est une liaison **série synchrone** (avec une ligne d'horloge entre deux circuits)
- Fonctionne en **full duplex** (Chaque circuits communicant peut en même temps émettre et recevoir des données).
- La communication est réalisée selon un schéma **maître-esclaves**, où le maître (généralement, un microcontrôleur ou une carte Arduino) contrôle la communication.
- Le bus SPI utilise quatre signaux logiques (figure II.12) :
    - **SCLK** : Serial Clock, Horloge (généré par le maître).
    - **MOSI** : Master Output, Slave Input (généré par le maître).
    - **MISO** : Master Input, Slave Output (généré par l'esclave).
    - **SS** : Slave Select, Actif à l'état bas (généré par le maître).

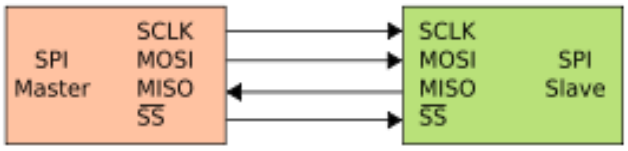


Figure II.12 Le bus SPI avec un maître et un esclave

- Plusieurs esclaves (exemples : Carte SD, EEPROM SPI) peuvent être reliés au même bus et la sélection du destinataire se fait par une ligne appelée **Slave Select SS** (figure II.13).

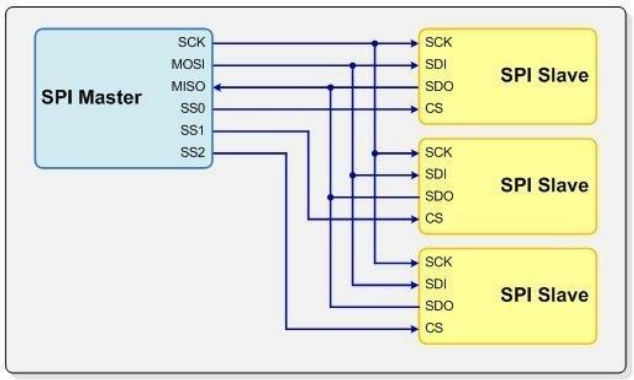


Figure II.13 Bus SPI avec un maître et plusieurs esclaves

Principe de mise en œuvre : La communication sur le bus est orchestrée de la manière suivante :

1. Le maître sélectionne l'esclave avec lequel il souhaite communiquer en mettant un niveau **bas** sur la ligne **SS** correspondante (Mettre la broche /SS à HIGHT signifie que le périphérique doit libérer le bus et rester silencieux).
2. Le maître génère le signal d'horloge
3. A chaque coup d'horloge, le maître et l'esclave s'échangent un bit sur les lignes MOSI et MISO selon le principe indiqué sur la (figure II.15).
4. Lorsque les 8 bits ont étés transmis, le maître arrête l'horloge et refait passer la ligne SS au niveau haut.

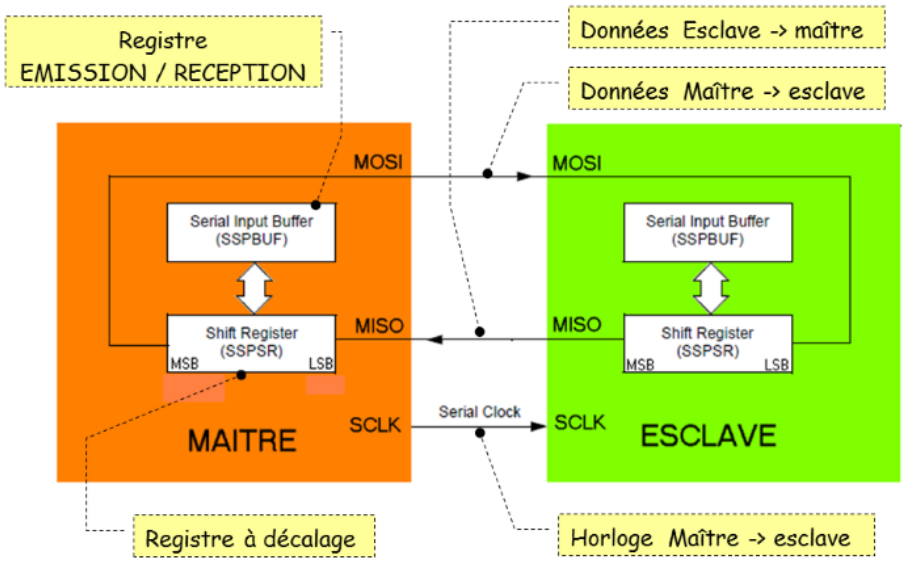


Figure II.14 Principe d'une communication SPI

Il y a trois paramètres de configuration de l'horloge :

- La fréquence d'horloge.
- La polarité de l'horloge, CPOL (Clock polarity)
- La phase de l'horloge, CPHA (Clock phase)

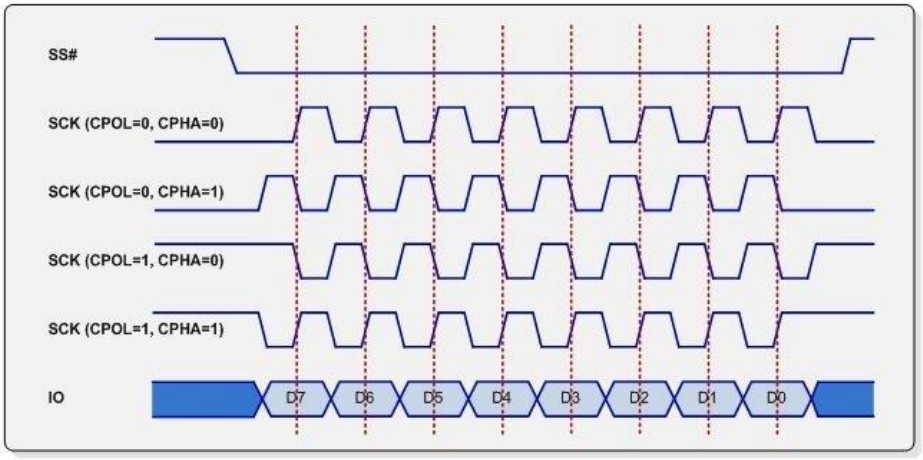


Figure II.15 Signal d'horloge dans les quatre modes de transmission

Il existe quatre modes de transmission (figure II.15) qui contrôlent si les données sont transmises sur le **front** montant ou descendant de l'horloge (appelé **phase** d'horloge), et si l'horloge est **à l'état de repos** lorsqu'elle est à **l'état haut ou bas** (appelée **polarité** d'horloge). Les quatre modes combinent la polarité et la phase selon ce tableau II.1.

Mode	CPOL	CPHA
0 (0,0)	0	0
1 (0,1)	0	1
2 (1,0)	1	0
3 (1.1)	1	1

Tableau II.1 Les quatre modes d'une transmission SPI

Un autre paramètre d'une transmission SPI est **l'ordre des données** transmises. Deux cas sont possibles : **MSB first** et **LSB first**.

En plus de la configuration standard à 4 fils de l'interface SPI, il existe une variante à 3 fils qu'on appelle **« Three wire »** (figure II.15) qui a but pour réduire le nombre de broche sur les composants (capteurs, mémoires, …). Dans cette configuration, une ligne de données joue tantôt le rôle de MOSI quand le composant doit recevoir des données ou des commandes du maître, et elle joue le rôle de MISO quand c'est le composant qui doit envoyer des données au maître. Comme on peut le voir, dans cette configuration les transactions sont semi-duplex

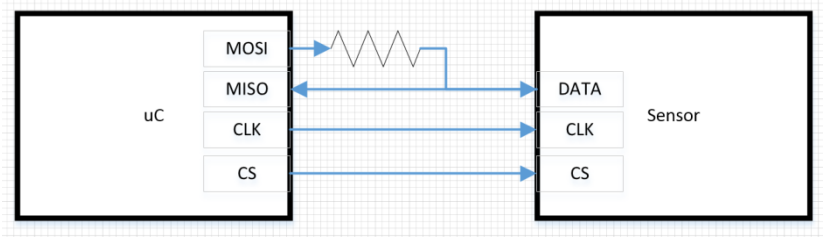


Figure II.16 La communication SPI de 3 fils

pour permettre une communication bidirectionnelle. La réduction du nombre de lignes de données diminue le débit maximal possible des appareils à 3 fils. La configuration **« Three wire »** est plutôt conçue pour les composants de faibles exigences de performances.

Dans une carte Arduino UNO, les quatre broches d'une liaison **SPI** sont 13 (SCLK), 12 (MISO), 11 (MOSI), et 10 (SS) (figure II.9). Si on a plusieurs esclaves sur le bus on peut prendre les autres lignes SS nécessaires parmi les lignes d'E/S de l'Arduino.

Dans l'ensemble de bibliothèques Arduino installées avec l'IDE Arduino, il y a une **bibliothèque SPI** qui permet de gérer une communication SPI entre la carte Arduino comme maître, et un autre circuit connecté.

Pour utiliser la bibliothèque SPI dans un programme, on doit écrire au début la directive : **`#include <SPI.h>`**.

Dans cette bibliothèque, **SPI** est l'objet représentant le bus SPI. Les principales fonctions associées à cet objet sont :

- **`SPI.beginTransaction(SPISettings(`**`MaxSpeed,dataOrder, dataMode`**`)`**

  La fonction **`beginTransaction`** a pour rôle d'initialiser et configurer le bus SPI. Elle admet comme seul paramètre l'objet **`SPISettings`** qui définit les configuration de la communication :
  - La vitesse max de l'horloge en hz
  - L'ordre des données: **`MSBFIRST`** or **`LSBFIRST`**
  - Le mode de la transmission : **`SPI_MODE0`**, **`SPI_MODE1`**, **`SPI_MODE2`**, ou **`SPI_MODE3`**

  Exemple :
  ```
  SPI.beginTransaction(SPISettings(14000000, MSBFIRST, SPI_MODE0))
  ```

- **`receivedVal = SPI.transfer(val)`**
- **`receivedVal16 = SPI.transfer16(val16)`**

  La fonction **`transfer()`** (ou **`transfer16()`**) est basée sur **l'envoie** et la **réception** simultanée de données.

  val est l'octet à envoyer sur le bus. **val16** est la valeur en deux octets à envoyer sur le bus. La donnée reçue est la valeur retournée par la fonction (**`receivedVal`** ou **`receivedVal16`**).

- **`SPI.endTransaction()`** : Arrêtez d'utiliser le bus SPI pour permettre à d'autres bibliothèques d'utiliser le bus SPI.

II.4 Bibliothèque Arduino ADNS5050

Dans notre circuit la souris, plus exactement son capteur ADNS-5050, est reliée à la carte Arduino UNO au moyen d'une liaison SPI three-wires. Pour lire les images capturées par le capteur de la souris ADNS-5050, on doit écrire un programme résident dans la carte Arduino qui va se charger de lire cette information. Ce programme consiste en une gestion de certains registres internes du capteur et la gestion de la communication SPI three-wires.

Vu que nous savons comment gérer ces registres pour lire les images capturées par la souris, et qu'il existe une bibliothèque SPI pour Arduino qui facilite la programmation de la communication entre l'Arduino et le capteur ADNS-5050, il nous ait possible d'écrire le programme qui réalise la tâche voulue.

Avant de nous lancer dans l'écriture de ce programme on s'est d'abord demandé s'il n'existe pas de bibliothèque Arduino pour ce capteur. Ce qui facilitera la programmation puisque cette bibliothèque gèrera en même temps les registres internes et la communication SPI. Dans l'IDE Arduino le « gestionnaire de bibliothèques » fournit une très grande liste de bibliothèques Arduino qui est constamment mise à jour, et à partir de laquelle on peut télécharger ces bibliothèques. On est donc allé vérifier dans cette liste s'il existe une bibliothèque pour le capteur ADNS-5050. Dans la liste il n'existe aucune bibliothèque pour ce capteur.

En cherchant sur internet on a trouvé qu'il existe deux bibliothèques Arduino pour le capteur ADNS-5050, l'une écrite en 2014 par « Hiroyuki Okada » sous le nom « ADNS5050 », et la deuxième écrite en 2018 par « Simone Cociancich » sous le nom « arduino-adns-5050 », toutes les deux téléchargeables à partir du site github **[w3][w4]**.

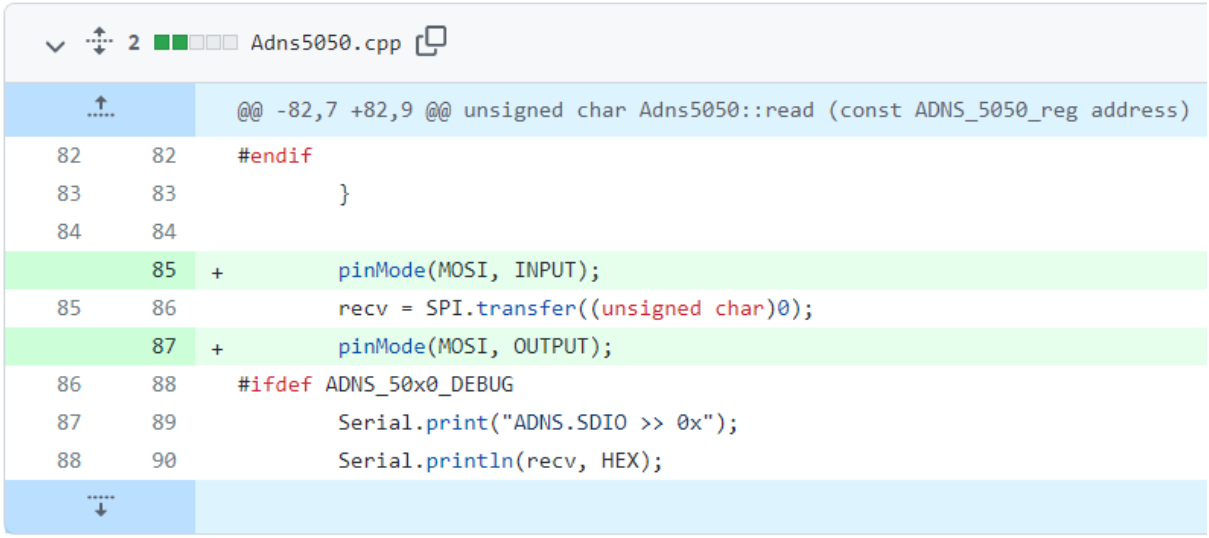
En jetant un coup d'œil sur les codes sources de ces deux bibliothèques on a remarqué que l'auteur de la 2ième bibliothèque a utilisé la bibliothèque Arduino SPI pour la communication, alors que l'auteur de la 1ière bibliothèque programme lui-même la communication SPI, peut-être parce que la bibliothèque SPI n'existait pas encore au moment où cette bibliothèque a été écrite. Ce qui fait que le code source de la 1ière bibliothèque est plus difficile à lire. En lisant le code de la 2ième bibliothèque on remarque qu'il concorde exactement avec le fonctionnement du capteur qui est décrit dans le datasheet. C'est pour cette raison qu'on a choisi d'utiliser cette deuxième bibliothèque.

Après avoir installé cette bibliothèque, et avant d'écrire notre programme, on a d'abord écrit un simple petit programme qui lit la valeur du registre Product_ID dont la valeur est connue (0x12) pour tester la bibliothèque. Seulement ce programme de teste n'a pas fonctionné. Le programme est trop simple et ne pouvait pas contenir d'erreur. Le problème ne pouvait venir que de la bibliothèque. En revenant dans la page github de cette bibliothèque on trouve un signalement d'erreur d'un utilisateur qui propose pour la réparer, d'ajouter deux lignes d'instructions dans le code de la bibliothèque. Ce n'est qu'en faisant cette réparation que la bibliothèque a fonctionné.

Cette erreur est due à un détail qui manque dans la bibliothèque. Ce détail est en rapport avec la configuration particulière du bus SPI (three-wire) dans laquelle les broches MOSI et MISO

du microcontrôleur sont toute deux reliées à la broche SDIO du circuit ADNS-5050. En effet, pour éviter tout conflit pendant l'opération de lecture on doit configurer la broche MOSI en entrée avant le transfert de la donnée, et on doit le remette en sortie après le transfert. C'est ces deux détails qu'il faut ajouter à cette bibliothèque (figure II.16).

Jusqu'au moment où on écrit ce mémoire l'auteur de la bibliothèque n'a pas encore réagit au signalement et n'a fait aucune réparation. Donc il faut faire attention à cela si vous devez installer cette bibliothèque.



2 Adns5050.cpp

```
@@ -82,7 +82,9 @@ unsigned char Adns5050::read (const ADNS_5050_reg address)
82  82    #endif
83  83          }
84  84
    85  +       pinMode(MOSI, INPUT);
85  86          recv = SPI.transfer((unsigned char)0);
    87  +       pinMode(MOSI, OUTPUT);
86  88    #ifdef ADNS_50x0_DEBUG
87  89          Serial.print("ADNS.SDIO >> 0x");
88  90          Serial.println(recv, HEX);
```

Figure II.17 La réparation de la bibliothèque Adns5050 à faire

Pour inclure cette bibliothèque on doit ajouter au début du programme la directive :

`#include "Adns5050.h"`

La bibliothèque « arduino-adns-5050 » contient trois fonctions :

`begin(freq)` : Cette fonction permet d'initialiser la communication SPI (fréquence, mode et ordre des données).

`data = read(Reg_adr)` : Cette fonction permet de lire la valeur (data) d'un registre interne de l'ADNS-5050 dont l'adresse est Reg_adr.

`write(Reg_adr,data)` : Cette fonction permet d'écrire la valeur « data » dans le registre dont l'adresse est Reg_adr

.

II.5 Traitement de correlation d'image avec OpenCV

II.5.1 Présentation de la bibliothèque OpenCV

Figure II.19 Logos d'OpenCV et Microsot Visual Studio

OpenCV (Open source Computer Vision) est une bibliothèque de programmation de vision par ordinateur en temps réel. Initialement développée par Intel en 1999. La société de robotique Willow Garage et la société ItSeez se sont succédé au support de cette bibliothèque. Depuis 2016 et le rachat d'ItSeez par Intel, le support est de nouveau assuré par Intel.

OpenCV est une bibliothèque libre et disponible pour la plupart des systèmes d'exploitation et existe pour les langages Python, C++ et Java. L'un des buts de cette bibliothèque est de fournir un moyen simple qui aiderait les utilisateurs à construire rapidement des applications assez sophistiquées. Cette bibliothèque contient plus de 500 fonctions qui couvrent plusieurs domaines de la vision artificielle dont, le contrôle de produits industriels, l'imagerie médicale, la sécurité des lieux, les multimédias, la robotique, etc.

Etant donné que la vision par ordinateur et le Machine Learning vont souvent de pair, OpenCV contient une bibliothèque de Machine Learning à usage générale (ML).
Dans notre cas on utilise OpenCV avec le langage C++ sous l'environnement de développement Visual Studio.

II.5.2 Mise en correspondance d'images (Template Matching)

Dans notre application on utilise le traitement de « template matching » sur les images capturées par la souris optique pour déterminer ses déplacements comme nous l'avons déjà cité au chapitre I. Dans cette partie on expliquera le template matching à travers un exemple qu'on rencontre souvent dans la littérature. Et on présentera la fonction OpenCV **`MatchTemplate()`** qui réalise cette tâche. L'exemple donné consiste à trouver le patch dans l'image d'entrée (l'image qui rassemble tous les robots) qui correspond le plus au modèle d'image (la petite image représentant le robot Wall.e).

Figure II.20 L'image template (à gauche), l'image d'entrée (à droite)

II.5.3 Fonction OpenCV qui réalise la mise en correspondance (Template matching) [d2]

```
void cv::matchTemplate(
  cv::InputArray  image,    // Input image to be searched, 8U or 32F, size W-by-H
  cv::InputArray  templ,    // Template to use, same type as 'image', size w-by-h
  cv::OutputArray result,   // Result image, type 32F, size (W-w+1)-by(H-h+1)
  int             method    // Comparison method to use
);
```

- **`image`** est l'image (W x H) d'entrée dans laquelle on doit chercher le modèle.
- Le modèle **`templ`** n'est qu'un patch (w x h) d'une autre image contenant l'objet recherché
- **`result`** est une matrice de dimension (W-w+1) x (H-h+1) des valeurs de corrélations calculées entre l'image modèle (template) et tous les patch de l'image d'entrée

- La méthode de mise en correspondance est choisie parmi l'une des options suivantes :
  - **Sum of Squared Differences** Matching Method (**`method = TM_SQDIFF`**)

$$R_{sq\_diff} = \sum_{x',y'} [T(x',y') - I(x+x',y+y')]^2$$

    Avec cette méthode, une correspondance parfaite donnera un résultat égal à **0**.

  - **Correlation** Matching Method (**`method = TM_CCORR`**)

$$R_{ccorr} = \sum_{x',y'} T(x',y') \cdot I(x+x',y+y')$$

    Avec cette méthode, le résultat **maximum** correspondra à une correspondance parfaite.

  - **Correlation Coefficient** Matching Method (**`method = TM_CCOEFF`**)

$$R_{ccoeff} = \sum_{x',y'} T'(x',y') \cdot I'(x+x',y+y')$$

$$T'(x',y') = T(x',y') - \frac{\sum_{x'',y''} T(x'',y'')}{(w-h)}$$

$$I'(x+x',y+y') = I(x+x',y+y') - \frac{\sum_{x'',y''} I(x'',y'')}{(w-h)}$$

    Avec cette méthode, une correspondance parfaite est égale à **1**

À chacune de ces méthodes il existe une version normalisée :

- Normalized Square Difference Matching Method (**`method = TM_SQDIFF_NORMED`**)
- Normalized Cross-Correlation Matching Method (**`method = TM_CCORR_NORMED`**)
- Normalized Correlation Coefficient Matching Method (**`method = TM_CCOEFF_NORMED`**)

Dans notre cas on utilise la méthode de mise en correspondance « Square Différence ».

Une fois qu'on utilise la fonction **`matchTemplate()`** pour obtenir l'image de correspondance **`result`**, on peut utiliser la fonction **`minMaxLoc()`** pour trouver la position de la meilleur correspondance.

On a remarqué dans nos simulations qu'on avait une détection plus précise lorsque le nombre de lignes et de colonnes de l'image template était impair.

II.5.3.1 Le programme de l'exemple donné

Dans ce programme la fonction **`matchTemplate ()`** retourne la matrice C des valeurs de corrélation entre l'image template « wali.jpg » et tous les patchs de même dimension de l'image « tous.jpg ». La valeur minimal correspond au patch qui semblable à l'image template. Pour localiser ce minimum on utilise la fonction **`minMaxLoc().`** Ensuite on utilise cette localisation et la fonction **`rectangle`** pour dessiner un rectangle autour du patch trouvé. La localisation du minimum correspond sur l'image « tous.jpg » au coin de limage situé en haut à gauche dans le patch trouvée.

```
3   using namespace std;
4   using namespace cv;
5   int main()
6   {
7       Mat A = imread("g:/tous.jpg");
8       Mat B = imread("g:/wali.jpg");
9       Mat C;
10      matchTemplate(A, B, C, TM_SQDIFF);
11
12      Point Pmin;
13      double min;
14      minMaxLoc(C, &min, 0, &Pmin);
15
16      rectangle(A, Pmin, Point(Pmin.x + B.cols, Pmin.y + B.rows), Scalar(0, 0, 255), 2.5);
17
18      namedWindow("fen1", 1);
19      namedWindow("fen2", 1);
20      imshow("fen1", A);
21      imshow("fen2", B);
22      waitKey();
23      return 0;
24  }
```

Figure II.20 Programme de l'exemple de template matching

II.6 Communication entre la carte Arduino et un programme qui tourne sur l'ordinateur

Le microcontrôleur d'une carte Arduino n'est pas dédié aux calculs volumineux avec contrainte de temps réel comme les calculs de traitement d'image. Ces traitements sont généralement effectués par des programmes écrit en langage C++ (ou java) et une bibliothèque de traitement d'image (comme OpenCV) exécutés par un microprocesseur assez puissant (comme le microprocesseur d'un PC).

Le programme sur la carte Arduino qui lit les images capturées par la souris ne traitera pas ces images. On a donc besoin d'une bibliothèque qui permet le transfert de ces images de la carte Arduino au programme de traitement d'image qui fonctionne sur le PC à travers une liaison série.

Pour permettre à la carte Aduino de transférer les images capturées par le capteur de la souris au programme de traitement en marche sur le PC, à travers le port série, on a utilisé la bibliothèque nommée **serialPort [w5] [v7]**. De façon générale, cette bibliothèque permet à un programme qui marche sur une carte Arduino et un programme qui marche sur le PC de s'échanger des données.

La bibliothèque **serialPort** est utilisée dans le programme en C++ qui s'exécute sur le PC. Elle permet à ce programme de communiquer avec une carte Arduino à travers le port série.

Cette bibliothèque définit une classe SerialPort avec quatre fonctions :

```
SerialPort(char *portName);
int readSerialPort(char *buffer, unsigned int buf_size);
bool writeSerialPort(char *buffer, unsigned int buf_size);
bool isConnected();
```

1. La fonction **`SerialPort()`** permet de définir une carte Arduino reliée au port série du PC (com1 ou com2 ou com3) comme un objet **`SerialPort`**.

2. La fonction **`isConnected()`** retourne la valeur logique true si la communication sur le port spécifié est opérationnelle. Elle retourne la valeur false si la connexion n'est pas établie. Dans ce cas, il se peut qu'il y ait une erreur sur le nom du port.

3. La fonction **`readSerialPort()`** permet au programme sur le PC de lire la chaine de caractère (1^{ier} paramètre de la fonction) envoyée par la carte Arduino.

4. La fonction **`writeSerial()`** permet au PC d'envoyer sur le port série spécifié une chaine de caractères (1^{ier} paramètre de la fonction) à la carte Arduino.

Pour utiliser cette bibliothèque dans une solution Visual studio il suffit d'ajouter à cette solution les fichiers **SerialPort.cpp** et **SerialPort.h**.

Remarques

1. Dans le programme, le nom du port de communication (COM1, COM2, etc.) est une chaine de caractère qu'on définit comme un tableau dynamique de caractères (ou un

pointeur sur un caractère). Par exemple pour le port COM3 le nom est exprimé dans le programme en C++ par quatre caractères antislash suivies d'un point suivi de deux caractères antislash suivis du nom du port en majuscule.

```
char *port = "\\\\.\\COM3";
```

2. Comme on peut le voir, les données échangées entre la carte Arduino et le PC sont considérées dans le programme comme des chaines de caractères.

II.7 Conclusion

On a présenté dans ce chapitre les moyens matériels et logiciels nécessaires pour déterminer les coordonnées (x,y) des déplacements d'une souris optique en se basant sur les images capturées. Les moyens matériels sont : Une souris optique (PLEOMAX MO-200) avec un capteur d'Avago (ex Logitech) ADNS5050, et une carte Arduino UNO. Les moyens logiciels sont : La bibliothèque Arduino « Adns5050 », un programme de traitement de "template matching" en utilisant la corrélation d'images avec la bibliothèque OpenCV, et une bibliothèque (serialPort) pour la communication entre la carte Arduino et le programme de traitement qui tourne sur l'ordinateur

Chapitre III

Programmes et résultats

III.1 Introduction

Rappelons que notre but est de réaliser le système de la figure II.1 qu'on a expliqué dans les deux chapitres précédents. En ce qui concerne le coté matériel, ce système est constitué d'une carte Arduino UNO reliée d'une part à une souris optique et d'autre part à un ordinateur par le port série. Pour le coté programme, il y a un programme sur la carte Arduino et un programme qui marche sur le PC.

La méthodologie suivie pour réaliser ce travail était de construire ce système de façon progressive. Et donc, on a divisé notre problème en trois sous problèmes :

1. Lire les déplacements de la souris et les images capturées par son capteur, et afficher leurs valeurs sur l'ordinateur.
2. Ecrire un programme de traitement des images capturées offline pour en déduire les déplacements de la souris
3. Ecrire un programme de traitement des images capturées cette fois-ci en ligne pour en déduire les déplacements de la souris

On présente dans ce qui suit les applications qui concernent ces trois sous problèmes.

III.2 Application 1

III.2.1 Circuit électronique

Comme on l'avait mentionné dans l'introduction de ce chapitre, dans cette application on voudrait :

1. Lire les déplacements de la souris et les afficher sur le « moniteur série » de l'IDE Arduino.
2. Lire les images capturées par la souris et afficher les valeurs de leurs pixels dans le « moniteur série » de l'IDE Arduino. On utilisera ensuite ces valeurs pour afficher sur l'ordinateur ces images et voir leurs apparences.

La figure III.1 montre comment on doit relier le capteur de la souris à la carte Arduino. Rappelons que le capteur de la souris optique et la carte Arduino communiquent entre eux par une liaison SPI three-wires (broches SDIO, SCLK, NCS). Les pins de l'alimentation et de la masse des deux cartes (carte Arduino et carte souris) doivent être reliés entre elles. La souris n'est plus reliée au PC par le connecteur USB, donc elle prend son alimentation de la carte Arduino. C'est la carte Arduino qui contrôle le reset du capteur de la souris, c'est pour cela que la broche NRESET du capteur est reliée à la carte Arduino.

Capteur	------	**Arduino**
VDD	------	5v
Ground	------	GND
SDIO	------	MISO, MOSI (pins 11 et 12)
SCLK	------	SCLK (pin 13)
NCS	------	SS (pin 10)
NRESET	------	pin 9

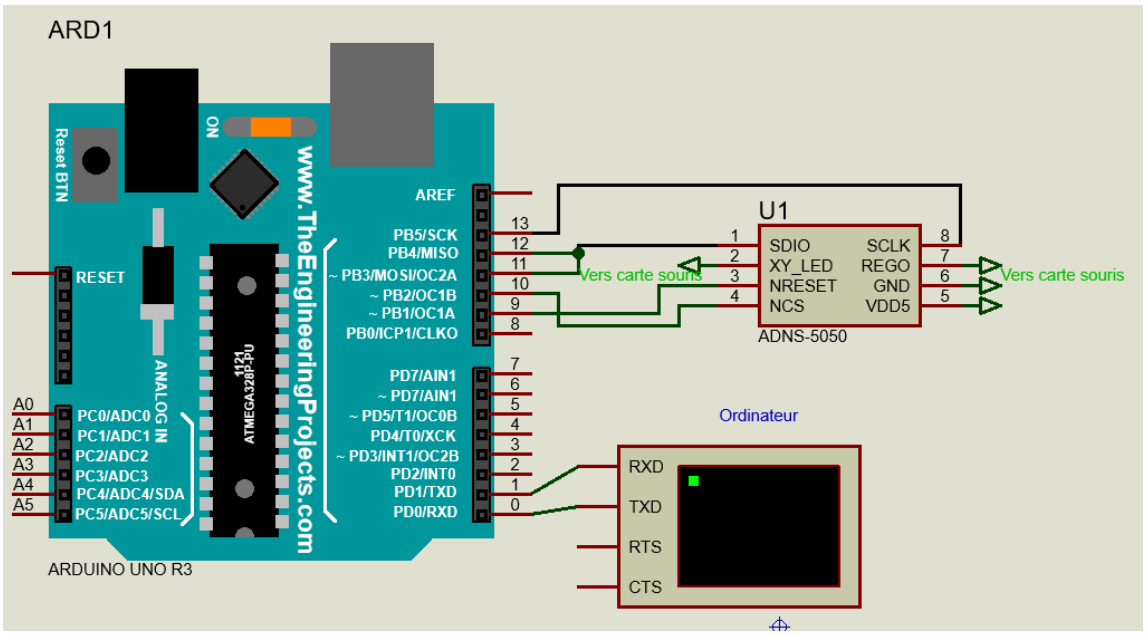


Figure III.1 Circuit électronique représentant les liaisons de la carte Arduino avec le capteur ADNS-5050 et le PC

Le circuit de la figure III.1 a pour seul but de montrer le schéma électrique et non d'en faire la simulation avec PROTEUS-ISIS car il n'existe pas dans la version présente d'ISIS un modèle de simulation du capteur ADNS-5050. En réalité il n'existe qu'un modèle schématique du capteur ADNS-5020 et qui a le même brochage que l'ADNS-5050. Dans la figure III.1 c'est le capteur ADNS-5020 qui est utilisé on lui a seulement changé de nom. La carte Arduino est aussi reliée au PC par le port USB.

Pratiquement, pour relier la carte Arduino au capteur de la souris qui est soudée sur la carte de circuit imprimé, on pourrait penser à trois différentes solutions :

- **Première solution** : Souder des fils sur les broches du capteur qui est encore soudé sur sa carte, et les relier aux pins correspondants dans la carte Arduino. En effet, il n'est pas question de dessouder le capteur de sa carte car rappelons que c'est ce capteur qui capture des images du plan de travail. C'est pour cela qu'il doit rester dans cette position.
- **Deuxième solution** : Dessouder le contrôleur de souris optique de la carte pour souder dans les trous qui le recevaient les fils qui vont relier le capteur de la souris à la carte Arduino. Car les pins du contrôleur sont dans certains cas directement reliés aux broches qui nous intéressent du capteur. Cette solution a été utilisé dans [w6] figure III.2.
- **Troisième solution** : Utiliser des fils à petites pinces pour relier la souris à la carte Arduino. Ces fils ne sont pas soudés sur les broches du capteur de la souris mais viennent seulement s'y accrocher.

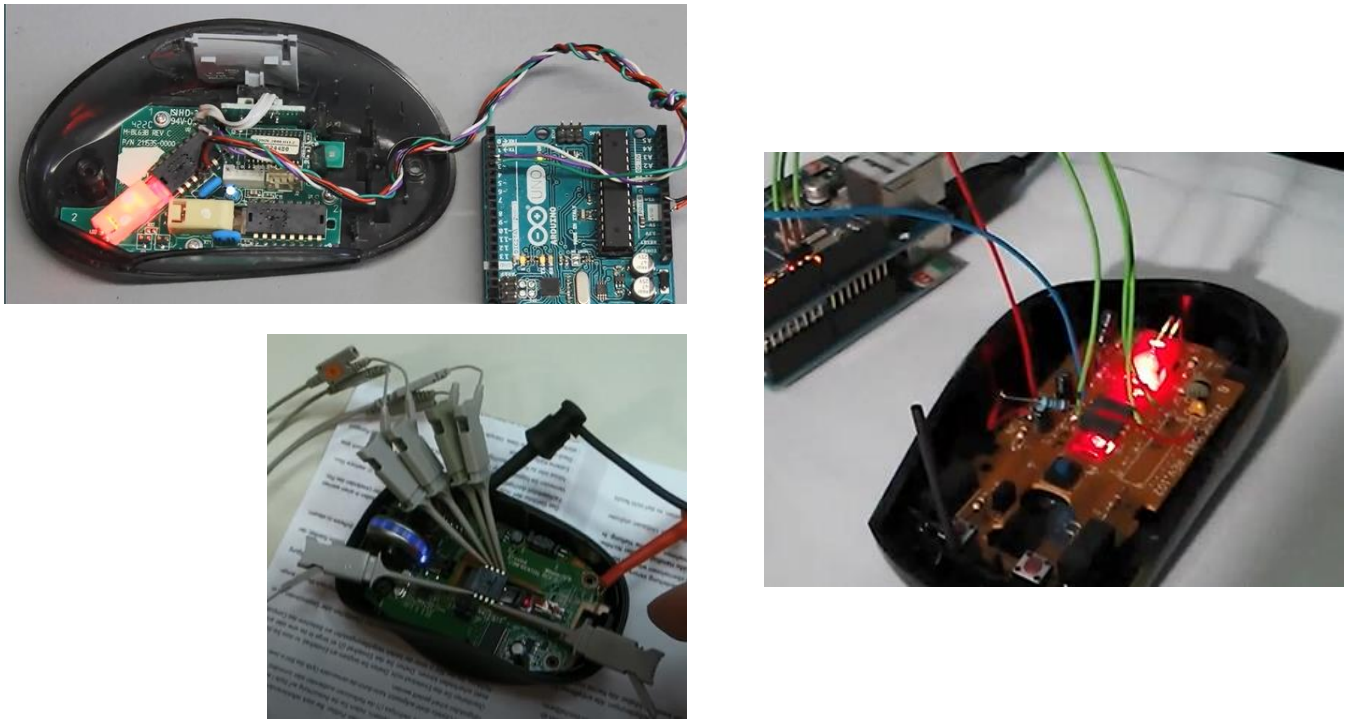


Figure III.2 Images montrant les trois différentes façons de relier une souris à une carte Arduino. Fils soudés sur les broches du capteur (**Image en haut à gauche**) – Fils soudés dans les trous du contrôleur de la souris après avoir dessouder ce dernier (**image à droite**) – Les broches du capteurs sont reliées à la carte Arduino avec des câbles à pinces (**image en bas à gauche**)

C’est la première solution qu’on a adopté au début. Mais il s’est avéré qu’elle n’est pas très pratique car d’un coté, ce n’est pas facile de souder les fils sur les broches du capteur à cause de l’encombrement sur la carte, et d’autre part, on a eu à beaucoup manipuler le circuit ce qui fait que les fils soudés finissaient par se détacher. Il fallait donc recourir à une autre solution. La solution deux qui est de dessouder le contrôleur de la souris n’est pas réalisable dans notre cas parce que certaines des broches du capteur qui nous intéressent ne sont pas directement reliées au contrôleur (elles sont reliées à travers un transistor) comme on peut le voir sur la figure II.2 qui représente le circuit de la carte de la souris. On a donc adopté la solution trois qui semble convenir le mieux.

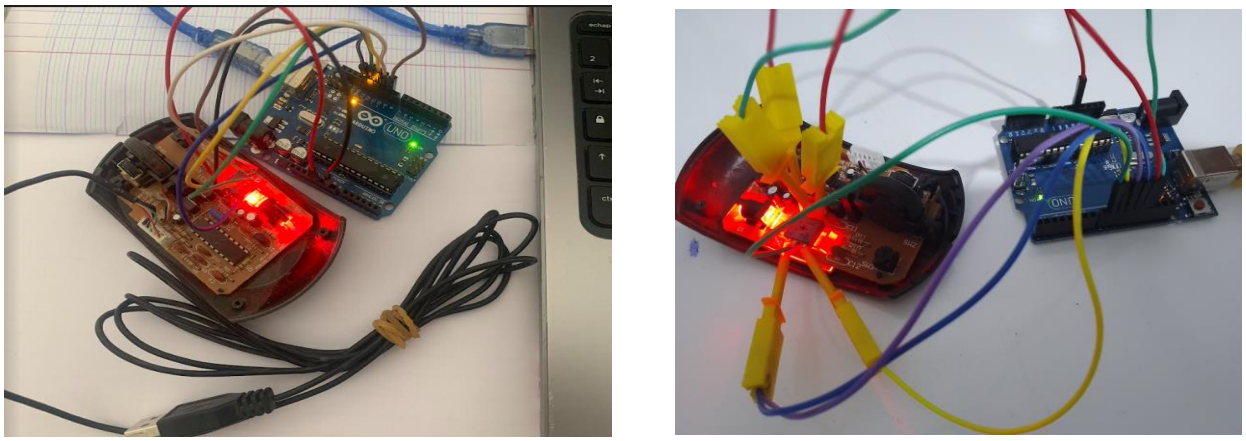


Figure III.3 Notre montage avec des câbles soudés sur les pins du capteur (à gauche) avec des câbles à pince (à droite)

III.2.2 Programme de teste

Après avoir téléchargé, corrigé (voir figure II.17) et installé la bibliothèque Arduino ADNS5050 on a écrit un petit programme pour tester le bon fonctionnement de cette bibliothèque. Le programme (figure III.4) consiste à lire la valeur du registre du capteur Product_ID et l'afficher dans le moniteur série de l'Arduino. D'après la liste des registres de l'ADNS-5050 de la figure I.2 ce registre ne peut pas être modifié et peut seulement être lu et a une valeur prédéterminée qui est 0x12 (18 en décimal). C'est bien la valeur qu'on voit affichée sur la console du moniteur série (figure III.5).

```
#include <Adns5050.h>
#include <adns50x0.h>

#define NCS 10   //Broche NCS sur la pin 10
#define NRESET 9 //Broche NRESET sur la pin 9

unsigned char val;
Adns5050 souris(NCS, NRESET);

void setup()
{
  souris.begin(1000000);
  Serial.begin(9600);

  val = souris.read(Product_ID);

  if (val==0x12)
  {
    Serial.print("Valeur du registre Product_ID (Hex) : ");
    Serial.println(val,HEX);
    Serial.println("La communication entre l'Arduino et l'ADNS-5050 a réussit");
  }
  else
  Serial.println("La communication entre l'Arduino et l'ADNS-5050 a échoué !");
}

void loop()
{
}
```

Figure III.4 Programme de teste : lit la valeur du registre Product_ID

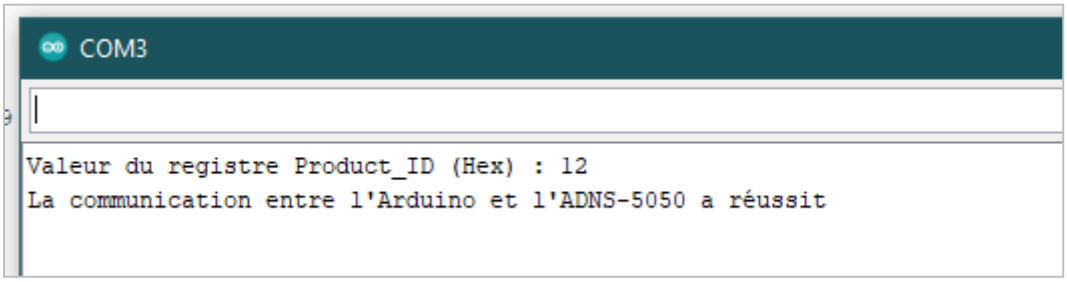


Figure III.5 La valeur du registre Product_ID s'affiche dans la fenêtre du moniteur série de L'Arduino

III.2.3 Programme qui lit les déplacements de la souris

On donne dans les figures III.6 et III.7 l'organigramme et le programme qui affiche les déplacements de la souris suivant X et Y déterminés par le DSP de l'ADNS-5050 après traitement des images capturées de la surface de travail. Toute nouvelle valeur de déplacements est sauvegardée dans les registres Delta_X et Delta_Y. D'après les informations données au chapitre I.4.5 sur les registres de l'ADNS-5050 on notera que :

- La valeur du registre « Motion » permet de détecter s'il y a eu un mouvement.
- Les valeurs des registres « Delta_X » et « Delta_Y » sont les déplacements suivant X et Y. Les valeurs de ces registres ne doivent être lues que si il y a une détection de mouvement (registre « Motion »).
- Un mouvement de la souris est détecté si le bit 7 du registre « Motion » est égal à 1. Autrement dit, la valeur de « Motion » est supérieure à 127.
- La valeur du déplacement est une valeur signée sur 8 bits. Donc c'est une valeur qui s'étend entre -128 et +127. En programmation une telle valeur correspond au type « char » contrairement à un octet non signés qui correspond au type byte (Arduino C).

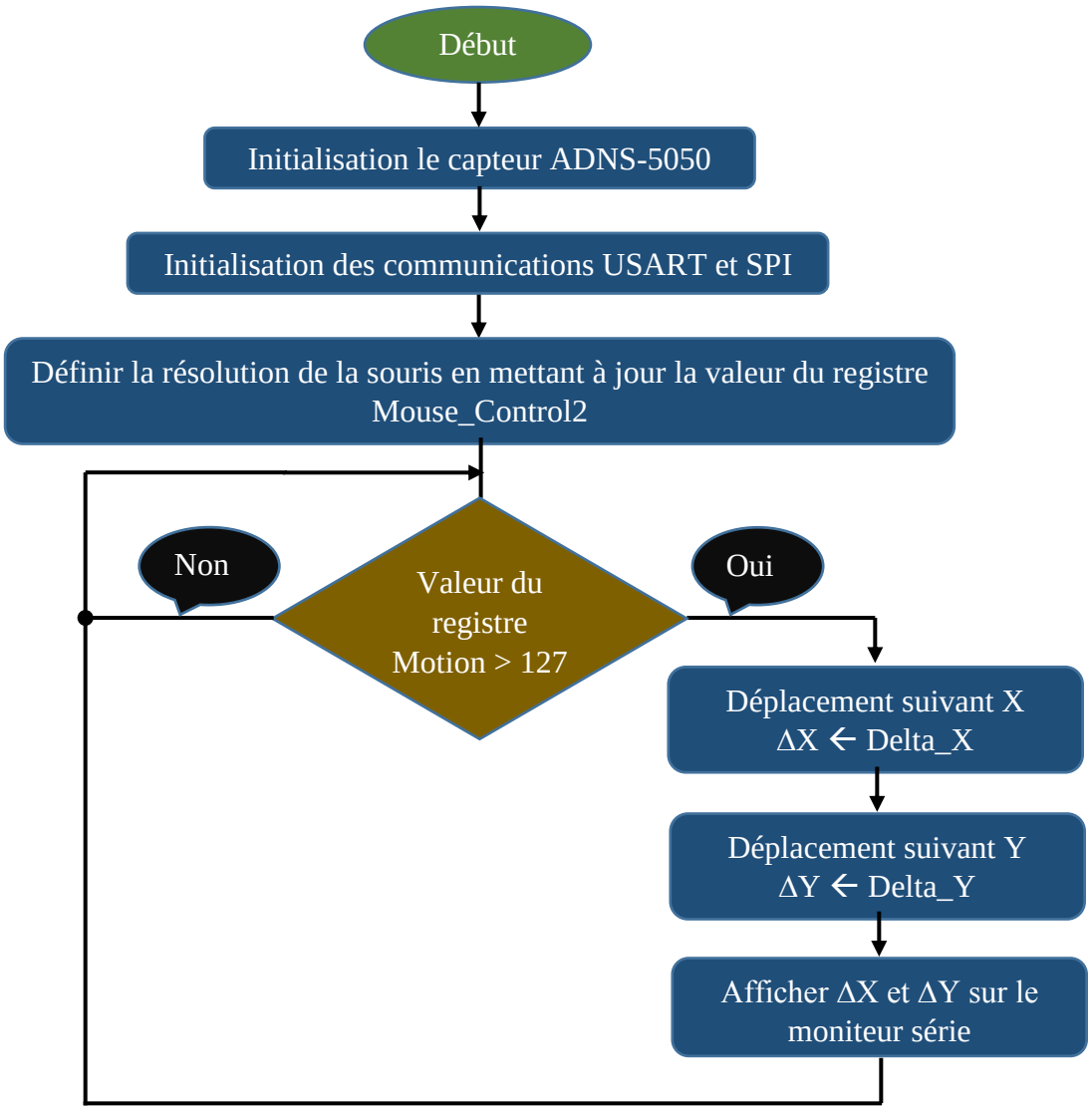


Figure III.6 Organigramme qui lit les déplacements de la souris

Prog2_App1 §

```
#include <Adns5050.h>
#include <adns50x0.h>

#define NCS 10   //Broche NCS sur la pin 10
#define NRESET 9 //Broche NRESET sur la pin 9

Adns5050 souris(NCS, NRESET);

void setup()
{
  souris.begin(1000000);
  Serial.begin(9600);
  souris.write(Mouse_Control2,B10110); //Résolution : 750 dpi
}

void loop()
{
  if (souris.read(Motion) > 127)
  {
    Serial.print("X = ");
    Serial.print(char(souris.read(Delta_X)),DEC);
    Serial.print("  ;  Y = ");
    Serial.println(char(souris.read(Delta_Y)),DEC);
  }
}
```

COM3

```
X = -58  ;  Y = 23
X = -50  ;  Y = 25
X = -38  ;  Y = 31
X = -25  ;  Y = 34
X = -16  ;  Y = 26
X = -12  ;  Y = 24
X = -10  ;  Y = 20
X = -6  ;  Y = 13
X = -3  ;  Y = 5
X = 2  ;  Y = 0
X = 0  ;  Y = -2
X = 0  ;  Y = -1
X = 1  ;  Y = 0
X = 0  ;  Y = -1
```

Figure III.7 Programme qui lit les déplacements de la souris (à gauche)
Affichage des valeurs de déplacement de la souris (à droite)

Résultats

On charge le programme de la figure III.7 sur la carte Arduino. Quand la souris est immobile aucune valeur de déplacement ne s'affiche sur le moniteur série. Quand on déplace la souris sur son plan de travail les valeurs des déplacements suivant x et y sont affichées. Plus vite on déplace la souris, plus les valeurs des déplacements sont grands.

III.2.4 Programme qui affiche les valeurs des pixels des images capturées

- Le registre Pixel_Grab contient initialement la valeur du 1^{ier} pixel. Quand on lit la valeur de ce registre il se charge automatiquement avec la valeur du pixel suivant. Une image capturée a en tout 361 pixels (19x19).
- L'image capturée par le capteur est une image en niveau de gris. Le capteur ADNS-5050 code la valeur d'un pixel sur 7 bits seulement.
- Le bit 7 du registre Pixel_Grab permet d'indiquer si la valeur qu'il contient est valide (bit 7=1 ==> Valeur valide). Donc avant de lire la valeur d'un pixel il faut d'abord vérifier que la valeur du registre Pixel_Grab est valide.
- Si le bit 7 indique que la valeur dans le registre n'est pas valide il faut forcer le registre pixel_Grab à pointer sur le premier pixel de l'image pour ensuite recommencer une

nouvelle lecture. On fait ceci en faisant une opération d'écriture dans le registre Pixel_Grab

- Chaque pixel lus sera affiché sur le moniteur série. Pour ne pas afficher les 361 valeurs sur une même ligne on ajoutera des instructions qui permettent d'avoir un affichage sur 19 lignes et 19 colonnes.

Les figures III.8 et III.9 montrent l'organigramme et le programme qui doit être chargé sur la carte Arduino.

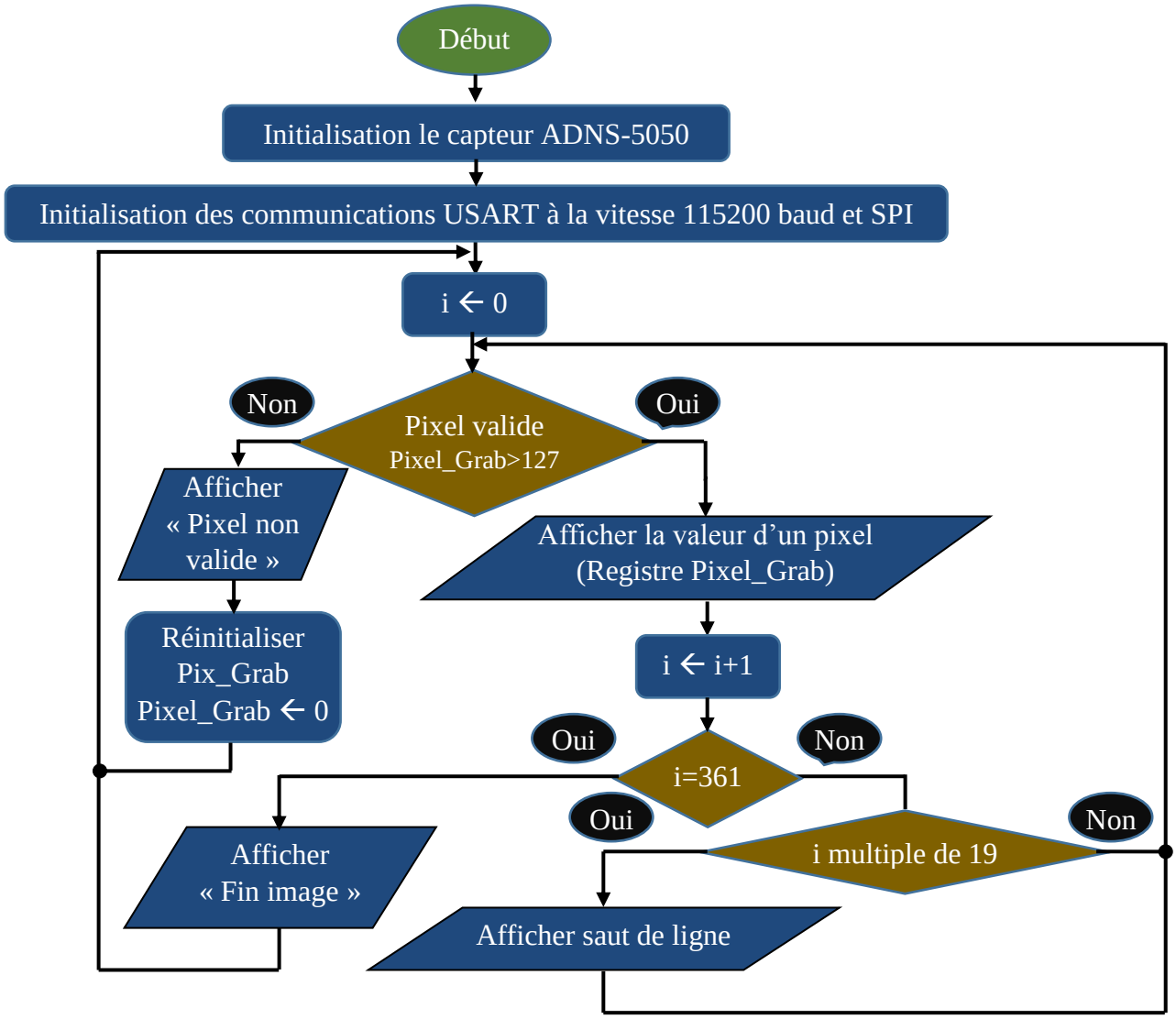


Figure III.8 Organigramme qui lit les pixels des images capturées et les affiche sur le moniteur série

Figure III. L'organigramme du programme d'affichage les valeurs de déplacements de la souris

```
#include <Adns5050.h>
#include <adns50x0.h>

byte pix;
int i = 0;
#define NCS 10   //Broche NCS sur la pin 10
#define NRESET 9 //Broche NRESET sur la pin 9

Adns5050 souris(NCS, NRESET);

void setup()
{
  souris.begin(1000000);
  Serial.begin(115200); //Vitesse max admise
  souris.write(Mouse_Control2, B10100);
  souris.write(Pixel_Grab, 0); //Forcer Pixel_Grab à lire le lier pixel
}

void loop()
{
  if (i % 19 == 0) Serial.println(); // Afficher chaque ligne d'image
                                     //dans une ligne à part
  pix = souris.read(Pixel_Grab);
  if (pix > 127) // Teste pixel valide
  {
    Serial.print(pix, DEC);
    Serial.print(" , ");
  }
  else
  {
    Serial.println("Valeur pixel non valide");
    souris.write(Pixel_Grab, 0); //Forcer Pixel_Grab à lire le lier pixel
    i = -1;
  }
  if (i < 360)
    i++;
  else
  {
    i = 0;
    Serial.println("STOP");
  }
}
```

Figure III.8 Programme qui lit les pixels des images capturées et les affiche sur le moniteur série

Pour afficher ces images capturées (figure III.11) on a écrit un petit programme en C++ et la bibliothèque (figure III.10). Les valeurs des pixels utilisées dans le programme ont été ajoutées dans le programme en copiant les valeurs des pixels qui sont affichées sur le moniteur série (figure III.9).

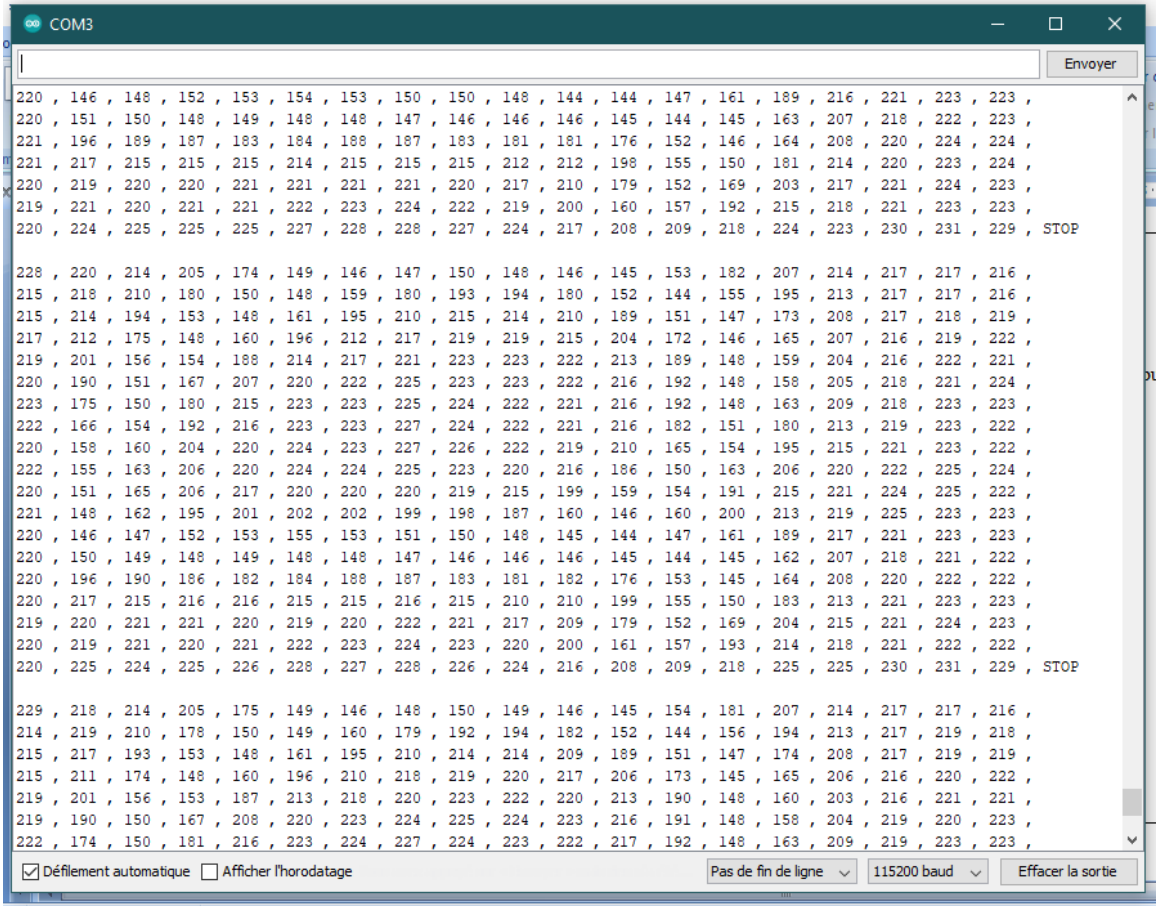


Figure III.9 Valeurs des pixels affichées sur le moniteur série

```
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;
int main()
{
    uchar B[361] = {
232 , 232 , 233 , 231 , 230 , 227 , 227 , 226 , 223 , 219 , 218 , 221 , 219 , 220 , 221 , 223 , 222 , 224 , 224 ,
223 , 233 , 232 , 231 , 228 , 226 , 224 , 218 , 214 , 209 , 210 , 213 , 211 , 216 , 220 , 222 , 222 , 222 , 224 ,
222 , 228 , 231 , 228 , 226 , 222 , 214 , 192 , 179 , 171 , 169 , 173 , 177 , 199 , 211 , 215 , 220 , 222 , 224 ,
221 , 233 , 230 , 228 , 221 , 203 , 178 , 154 , 150 , 148 , 146 , 146 , 147 , 158 , 179 , 205 , 218 , 219 , 223 ,
220 , 235 , 229 , 224 , 205 , 167 , 152 , 147 , 149 , 149 , 152 , 150 , 145 , 146 , 150 , 181 , 208 , 217 , 219 ,
219 , 235 , 228 , 218 , 180 , 152 , 151 , 155 , 173 , 183 , 190 , 187 , 172 , 148 , 145 , 156 , 191 , 214 , 219 ,
215 , 235 , 226 , 208 , 166 , 149 , 155 , 186 , 206 , 211 , 213 , 211 , 201 , 163 , 144 , 148 , 173 , 208 , 213 ,
212 , 233 , 225 , 191 , 156 , 148 , 169 , 204 , 215 , 219 , 219 , 217 , 209 , 184 , 148 , 146 , 161 , 198 , 205 ,
202 , 233 , 222 , 184 , 154 , 149 , 174 , 209 , 217 , 221 , 220 , 219 , 213 , 194 , 153 , 145 , 153 , 187 , 189 ,
169 , 230 , 223 , 183 , 152 , 150 , 179 , 211 , 218 , 219 , 217 , 218 , 213 , 199 , 159 , 143 , 150 , 175 , 161 ,
145 , 231 , 222 , 187 , 154 , 150 , 174 , 207 , 217 , 218 , 218 , 217 , 211 , 193 , 154 , 143 , 145 , 150 , 144 ,
143 , 232 , 226 , 200 , 158 , 152 , 156 , 193 , 209 , 213 , 213 , 212 , 200 , 171 , 148 , 142 , 143 , 143 , 144 ,
154 , 234 , 229 , 216 , 177 , 152 , 150 , 161 , 182 , 193 , 198 , 195 , 169 , 148 , 145 , 144 , 144 , 147 , 159 ,
186 , 234 , 230 , 223 , 202 , 166 , 149 , 148 , 152 , 153 , 155 , 157 , 147 , 144 , 145 , 147 , 154 , 179 , 197 ,
209 , 234 , 231 , 227 , 220 , 195 , 167 , 153 , 150 , 147 , 147 , 147 , 148 , 148 , 158 , 172 , 188 , 205 , 208 ,
213 , 227 , 224 , 224 , 222 , 215 , 206 , 189 , 177 , 162 , 158 , 157 , 167 , 177 , 191 , 203 , 209 , 213 , 215 ,
216 , 227 , 227 , 225 , 222 , 220 , 217 , 212 , 208 , 202 , 200 , 201 , 204 , 209 , 211 , 213 , 217 , 219 , 218 ,
217 , 227 , 227 , 226 , 224 , 223 , 219 , 218 , 216 , 213 , 211 , 213 , 215 , 215 , 216 , 218 , 218 , 220 , 220 ,
217 , 230 , 230 , 230 , 227 , 227 , 225 , 222 , 222 , 219 , 218 , 221 , 221 , 221 , 222 , 223 , 227 , 227 , 227 };
    Mat A(19, 19, CV_8UC1, B);
    Mat N;
    normalize(A, N, 0, 255, NORM_MINMAX);
    namedWindow("Image", WINDOW_KEEPRATIO);
    imshow("Image", N);
    waitKey();
    return 0;
}
```

Figure III.10 Programme qui lit les pixels des images capturées et les affiche sur le moniteur série

Comme l'éclairage est très fort et très proche de la surface les images obtenues sont très claires comme on peut les voir dans les valeurs qui sont affichées sur le moniteur série. Pour obtenir des images meilleurs on met à l'échèle les valeurs des pixels de l'image dans l'intervalle [0,255] en utilisant la fonction d'OpenCV normalize().

Résultats

Comme l'image est très petite (19x19) on a fait un agrandissement des images où un petit carré représente un pixel. On a fait l'expérience sur deux types de surfaces, surface très lisses et surface moins lisses. On choisi comme exemple de surface lisse les surface de couvertures de livres avec de très petites inscriptions pour obtenir des images contenant des formes reconnaissables. Et comme surface moins lisse une feuille de papier blanc.

Comme on peut le constater l'image de la surface de papier blanc est celle des irrégularités de sa surface. Contrairement aux surfaces lisses qui ne contiennent pas d'irrégularités et où on peut voir clairement les petites inscriptions sur la surface.

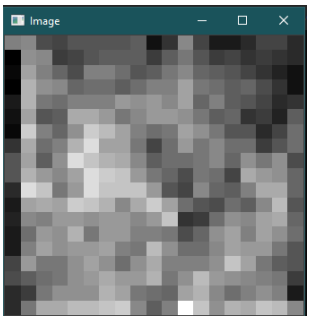


Figure III.11.a Image d'une surface de papier blanc

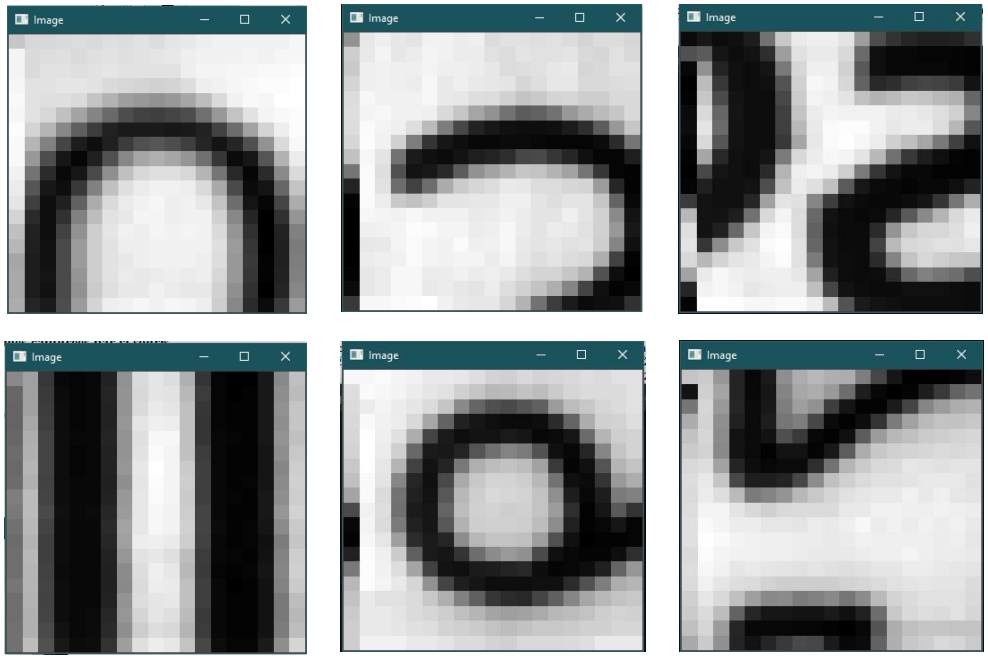


Figure III.11b Image d'une surfaces lisses avec de petite inscriptions

III.3 Application 2

Dans cette partie on va faire le traitement de corrélation (template matching) de deux images prises pendant le déplacement de la souris pour déterminer le déplacement de la souris.

III.3.1 Programme qui détermine le déplacement de la souris

L'organigramme suivant résume les différentes étapes du programme (figure III.13) pour déterminer le déplacement de la souris.

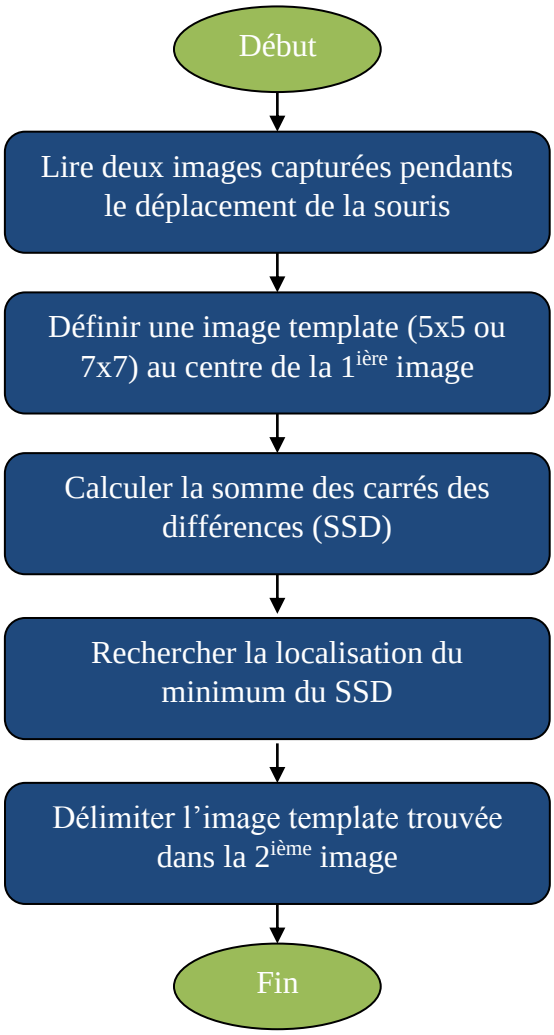


Figure III.12 Organigramme de la détection du déplacement de la souris

```cpp
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;
int main()
{
    Mat A = imread("image1.png");  // Image capturée initialement
    Mat B = imread("image2.png");  // Image capturé après déplacement de la souris
    Mat temp = Mat(A,Range(6,12),Range(6,12)); //Image template 7x7 partie centrale de l'image A
    Mat C;

    matchTemplate(B, temp, C, TM_SQDIFF);

    Point Pmin;
    double min;
    minMaxLoc(C, &min, 0, &Pmin);

    rectangle(B, Pmin, Point(Pmin.x + temp.cols, Pmin.y + temp.rows), Scalar(0,255,0), 0.25);
    rectangle(A, Point(6,6), Point(12,12), Scalar(0,255,0), 1);

    int delta_x = Pmin.x - 6; // Déplacement suivant x
    int delta_y = Pmin.y - 6; // Déplacement suivant y

    namedWindow("fen1", 2);
    namedWindow("fen2", 2);
    imshow("fen1", A);
    imshow("fen2", B);
    cout << "delta_x = " << delta_x << endl;
    cout << "delta_y = " << delta_y << endl;

    waitKey();
    return 0;
}
```

Figure III.13 Programme de la détection du déplacement de la souris

Résultats

On peut voir sur les images de la figure III.14 les résultats des traitements pour la détermination du déplacement dans le d'images lisses et moins lisses, et de différentes tailles de l'image template.

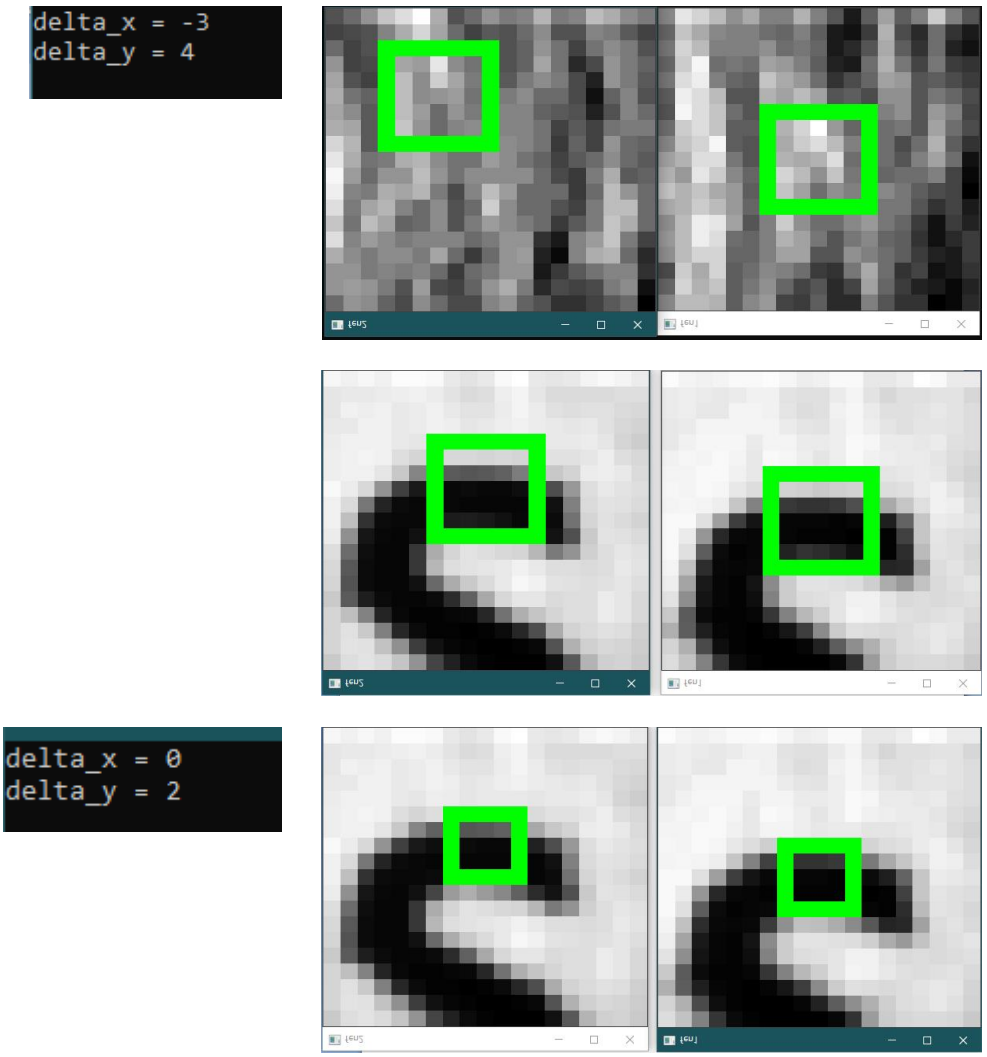


Figure III.14 Résultat dans les cas de surfaces lisses (les quatre images en bas) et moins lisses (les deux images en haut) en utilisants des templates de différentes tailles (5x5 et 7x7)

III.3 Application 3

Avant d'utiliser la bibliothèque SerialPort pour la transmission des images lues du capteur au programme de traitement sur le PC, on a fait un petit programme simple de teste.

Dans ce teste le programme de la carte Arduino (figure III.16) envoie en boucle sur le port série la valeur d'une variable qu'il incrémente à chaque itération.

Le programme sur le PC (figure III.15) doit lire la valeur reçue avec la fonction readSerialPort() de la bibliothèque SerialPort. Comme la valeur lue par cette fonction est de type tableau de caractères on doit la convertir en entier en utilisant la fonction stoi(). Et pour

vérifier la chaine de caractères a bien été transformé en entier on lui ajoute 10 et l'affiche sur le consol Window.

```
#include <iostream>
#include <opencv2/opencv.hpp>
#include "SerialPort.h"
#include <string>
#include <stdlib.h>

using namespace std;
using namespace cv;
char port[9] = "\\\\.\\COM3";
char pin[MAX_DATA_LENGTH];
String m;
int p=0;

int main()
{
    SerialPort arduino(port);
    if (arduino.isConnected())
        cout << "Connexion etablie" << endl;
    else
        cout << "erreur de connexion" << endl;
    while (arduino.isConnected())
    {
        arduino.readSerialPort(pin, MAX_DATA_LENGTH);
        m = String(pin);
        p=stoi(m);
        p += 10;
        cout << p << endl;
    }

    return 0;
}
```

Figure III.15 Programme qui fonctionne sur le PC

```
int i=0;
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.println(i);
  i++;
}
```

Figure III.15 Programme de la carte Arduino

Malheureusement ce teste n'a pas bien fonctionné. Il s'affiche quelques fois une succession juste de nombres et quelquefois des valeurs aléatoires. On a voulu essayé avec un teste encore plus simple dans lequel le programme de la carte Arduino envoie une valeur une seule fois et le programme sur le PC doit seulement la lire et l'afficher sans aucune conversion. Quand on a exécuté ce programme la valeur envoyée s'affiche plusieurs fois alors qu'on l'a envoyé qu'une seul fois.

On en déduit que cette bibliothèque sur laquelle on comptait faire le traitement des images en ligne ne fonctionne pas bien. Et malheureusement on n'a pas eu le temps de chercher et d'essayer d'autres moyens.

III.4 Conclusion

On a présenté dans ce chapitre les programmes suivants : Un programme qui lit les déplacements de la souris, un programme qui lit les images capturées par le capteur et affiche les valeurs de leurs pixels sur le moniteur série, et un programme de traitement d'image hors ligne pour déterminer les déplacements de la souris. Pour déterminer les déplacements de la

souris en ligne il faut juste ajouter au programme précédent les instructions de la bibliothèque seialPort qui lui permettent de lire les images envoyées par la carte Arduino afin de les traiter. Malheureusement, on a découvert à travers un simple programme de teste (lecture d'une seule valeur), que cette bibliothèque ne fonctionne pas bien pour on ne sait quelle raison. À cause du manque de temps, on n'a pas eu le temps de chercher et d’essayer une autre bibliothèque.

Conclusion général

En conclusion on estime qu'on a atteint les objectifs qu'on s'était donné au départ. Car on a réussi à lire les images capturées par la souris, et on a écrit un programme de traitement des images pour déterminer les déplacements de la souris. Un programme propre à nous qui traites des données brutes (les images capturées) sans recours aux traitements que fait le DSP grâce à ses programmes intégrés.

On aurait voulu faire ces traitements en ligne mais malheureusement la bibliothèque de communication entre la carte Arduino et le PC que nous envisagions d'utiliser ne fonctionne pas bien.

Ce projet nous a été très bénéfique car il nous a permis de découvrir cet appareil et d'acquérir de nouvelles connaissances en électronique et traitement d'image.

Référence

Docments

[d1] CMOS DIGITAL OPTICAL NAVIGATION CHIP - **Rajeev Badyal, Mark Alan Anderson, Brian James Misek – Brevet d'invention numéro : US 6,631,218 B2, 7 Octobre 2003**

[d2] NAVIGATION SYSTEM AND NAVIGATION METHOD - **Bang Won Lee & Jong Taek Kwak - Brevet d'invention numéro : US 2005/0060668 A9 , 17 Mars 2005**

[d3] Learning OpenCV 3 - **Adrian Kaehler & Gary Bradski – publié par O'Reilly Media en 2017**

[d4] Datasheet : ADNS-5050 – **Avago Technologies – 25 Avr 2012**

[d5] Datasheet : HT82M28A 3/5-Key USB Vista Tilt Optical Mouse Controller – **HOLTEK - 15 Avr 2008**

Sites web

[w1] History of the Mouse – **ProEdit –** https://proedit.com/history-of-the-mouse/

[w2] "Arduino" sur l'encyclopédie Wikipédia fr – dernière mise à jour 20/06/2022 - https://fr.wikipedia.org/wiki/Arduino

[w3] Arduin library "arduino-adns-5050" sur le site github - **Simone Cociancich (pseudonyme: shb) – 26 Nov 2017** - https://github.com/shb/arduino-adns-5050

[w4] Arduin library "ADNS5050 " sur le site github - **Hiroyuki Okada (pseudonyme: okhiroyuki) – 28 Janv 2015** - https://github.com/okhiroyuki/ADNS5050

[w5] "Connect-And-Use-Arduino-via-Cpp-Software-Made-In-Any-IDE" sur le site github – **ZainUlMustafa – 15/08/2017** - GitHub - ZainUlMustafa/Connect-And-Use-Arduino-via-Cpp-Software-Made-In-Any-IDE: Learn to make use of Arduino using your software designed in C++, Python, or Java in your electronic projects.

[w6] Optical Mouse Hacking - **Posted 30th March 2013** - Optical Mouse Hacking | WildCircuits

Vidéos

[v1] How does a Mouse know when you move it? || How Does a Computer Mouse Work? - **Branch Education - 16/10/2021** https://www.youtube.com/watch?v=SAaESb4wTCM

[v2] How Does a Mouse Work? - Basics Explained, H3Vtux-31/03/2019 https://www.youtube.com/watch?v=eccSwn9QVxo

[v3] How does an optical mouse work ?! - **Curious Owl – 24/11/2021**
https://www.youtube.com/watch?v=mLeGQtVqyr0

[v4] Décortiquer une souris optique - **Deus Ex Silicium – 5/11/2014**
https://www.youtube.com/watch?v=9nM18t2TxQQ

[v5] Voir à travers le capteur d'une souris optique - **Deus Ex Silicium – 07/05/2015**
https://www.youtube.com/watch?v=6Rg3LWxZicc

[v6] What does a computer mouse see? - **The 8-Bit Guy – 30/07/2020**
https://www.youtube.com/watch?v=xWB9dP1AtDU

[v7] Connect And Use Arduino Via Serial Library In C++ Software | Works With Any IDE! - **Sciengit - 19/09/2017** https://www.youtube.com/watch?v=8BWjyZxGr5o