HELHa
Haute École Louvain
en Hainaut

Projet et bureau d'étude
ScanBook

Rapport final

Wesley Estievenart – Thomas Herpoel – Michael Manzella

Table des matières

Introduction

Ce rapport présente les modifications apportées sur le projet de scanner de livres réalisé l'année précédente. Les nouveautés du prototype de cette année sont la construction d'une nouvelle structure et d'un nouveau système de traitement d'images.

Contexte

Le but de ce projet est de pouvoir scanner des livres sans abimer la structure de ceux-ci. L'utilisation d'une photocopieuse classique ne récupère pas des images de bonne qualité sur ce genre de support et peut abimer la reliure du livre.

Ce projet étant une demande de la bibliothèque de la HELHa, nous devons répondre aux besoins de cette dernière afin de scanner un livre de la manière la plus simple possible et sans l'abimer.

Les facteurs à retenir lors de la conception de notre prototype étaient la simplicité d'utilisation et la qualité des images.

Cahier des charges du client

Le cahier des charges qui nous a été remis en début de projet était le suivant. Nous devions réaliser un bilan sur l'état de fonctionnement de la structure actuelle. Nous devions aussi choisir une nouvelle caméra pour remplacer la webcam présente sur le prototype et qui était de très faible résolution. Nous devions aussi améliorer la partie software. Nous sommes passés d'un système contrôlé par un ordinateur sous Windows à un Raspberry Pi sous Raspbian. Ce dernier a pour but de rendre le système autonome et a comme fonction le traitement des images ainsi que la séquence du scanner. Lorsque le traitement est terminé, les images sont compilées dans un fichier PDF et ce fichier est envoyé par mail et/ou stocké dans un répertoire sur le Raspberry.

Le cahier des charges du client fourni au début du projet est repris ci-dessous.

- Faire le bilan sur l'état de fonctionnement et la reprise en main du prototype.
- Choisir une caméra haute résolution répondant aux contraintes du projet
- Test de la caméra indépendamment du système
- Amélioration de la partie software du scan (OCR,…)

  Intégration de la caméra dans le système

  Intégration des traitements sur la Raspberry Pi (opération de scan + transmissions des fichiers par mail ou sur carte SD)
- Améliorations mécaniques si nécessaire

Bilan et cahier des charges fonctionnel

Le projet existant est une preuve de concept et a rempli ce rôle de façon satisfaisante. Cependant, plusieurs éléments limitant ne permettent pas au projet existant de remplir son rôle de scanneur semi-automatisé de livre. Une liste de certains de ces éléments est donnée ci-dessous.

- La résolution de la prise de vue est loin d'être suffisante pour obtenir un document d'une qualité satisfaisante.
- Le traitement software effectué en environnement Processing ne permet pas d'exploiter la librairie OpenCV à son plein potentiel et limite les possibilités du système.
- Le système n'est pas du tout autonome et requiert l'utilisation d'un PC en continu.
- La mécanique du prototype n'est pas stable, elle n'est pas non plus résistante et n'est pas facilement modifiable.
- La carte électronique pilotant la motorisation entraine parfois la destruction du module de contrôle du moteur pas-à-pas.

Au vu de ces points, il est désormais temps d'orienter le projet vers un prototype fonctionnel et utilisable. Pour cela, l'équipe de projet s'est concertée de façon exhaustive afin de traduire les besoins du client en un cahier des charges fonctionnel. Ce cahier des charges sera présenté dans plusieurs sections. La première sera consacrée à l'architecture générale du système. Ce projet pouvant être défini au travers de trois grands axes (sa mécanique, son électronique et son software), chacune de ces parties sera détaillée dans une section également.

Architecture générale

- Utiliser un Raspberry Pi 3.
    - C'est le dernier modèle disponible du Raspberry Pi et le gain de puissance que cela implique par rapport à des systèmes précédents sera un plus pour le traitement des images.
- Utiliser la Pi camera 1.3 dans un premier temps car elle est déjà disponible, puis orienter les tests sur la caméra pi 2.1 une fois qu'elle sera commandée et reçue.
    - La Pi camera 2.1 offre une résolution de 8MP, ce qui est supérieur aux 5MP de la version 1.3.
- Utiliser 2 prises de vue, une pour chaque page.
    - Les résultats du projet en 2015-2016 avaient montré qu'une résolution de 20MP était conseillée pour obtenir une qualité satisfaisante. Les 16 MP offerts par ces 2 prises de vue se rapprochent des 20MP conseillés, surtout si l'on considère que les pages du livre seront mieux cadrées par les caméras et que donc moins de pixels seraient « gaspillés ».
- Utiliser un écran LCD embarqué sur le Raspberry Pi.
    - Permettra d'afficher l'interface graphique permettant d'utiliser le scanner en mode semi-automatique.
    - Le LCD choisi est un écran de type Waveshare 3.5 pouce.

Software

- Utiliser OpenCV pour effectuer le traitement d'image et la reconnaissance de pages.
    - C'est une librairie open source très documentée et ayant une grande communauté d'utilisateurs.
    - Doit être capable de reconnaitre la page, de la recadrer et de la redresser.
- Création d'un fichier PDF avec les pages scannées.
- Fonctionnalités : envoi du fichier PDF par mail, sauvegarde sur SD ou sauvegarde sur clé USB.
- Coder en C++.
    - Cela permet de rechercher le maximum de performances.

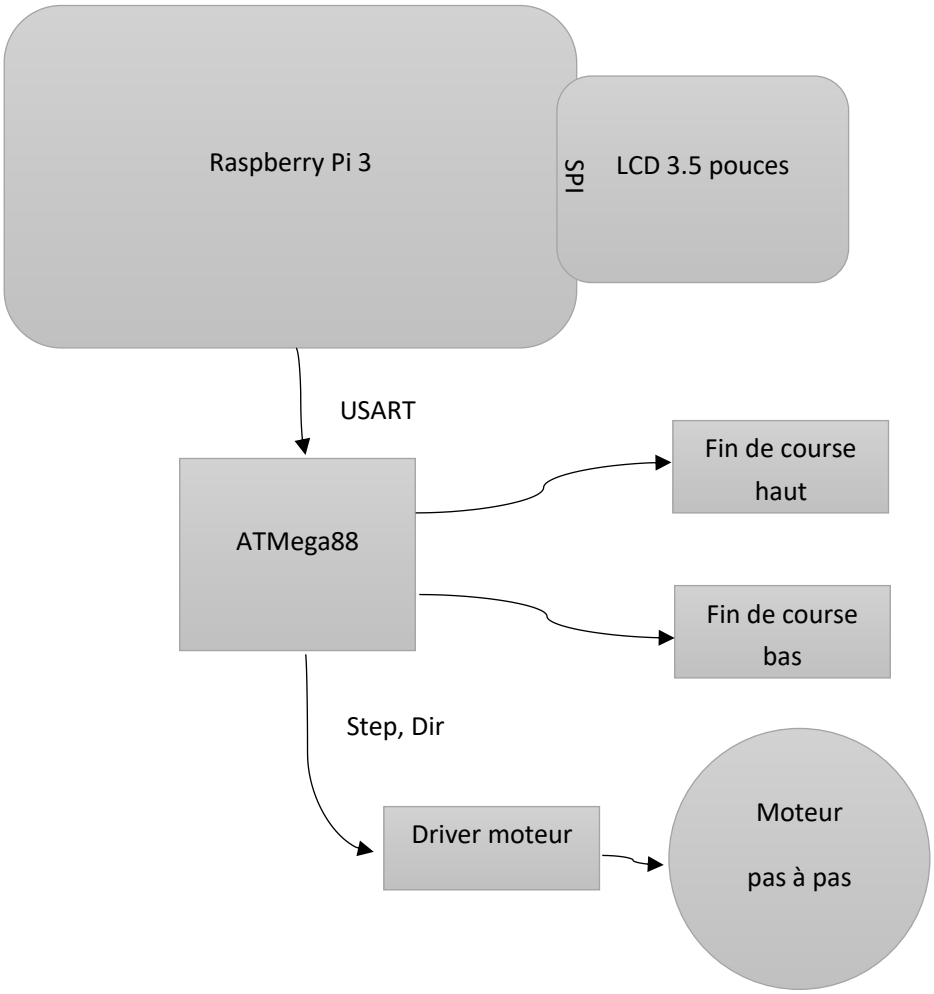
Electronique

- Concevoir un circuit électronique pour connecter deux Pi camera 2.1 au Raspberry Pi 3 afin d'effectuer les deux prises de vue (une pour chaque page).
    - Une alternative à ce système est d'utiliser une caméra montée sur un mobile et de faire mouvoir cette caméra pour faire les deux prises de vue. Cependant, le système utilisant deux caméras possède l'avantage de nécessiter moins de complexité mécanique.
- Réutiliser la platine électronique de 2015-2016 mais la modifier pour s'assurer de son bon fonctionnement.
- La gestion de la motorisation sera effectuée par un microcontrôleur relié au Raspberry Pi par un port série.
    - Le code de 2015-2016 sera modifié et simplifié pour être mieux adapté à la nouvelle version du projet.

Mécanique

- Les caméras sont fixées au sommet de la structure.
    - Cela permet d'éviter un câblage mobile.
- La structure est modulable.
- La structure est potentiellement fermée.
    - Cela permet par exemple de mieux maitriser la lumière.
- Réutiliser le support de livre de l'année précédente pour ce prototype.

Schéma bloc


Raspberry Pi 3
SPI
LCD 3.5 pouces
USART
ATMega88
Fin de course haut
Fin de course bas
Step, Dir
Driver moteur
Moteur pas à pas

Implémentation

Partie mécanique

Nous avons effectué une refonte complète de la structure afin de corriger certains points de la structure précédente (voir Figure 1). D'un point de vue esthétique, nous voulions modifier la structure, qui était essentiellement en bois, par une structure plus solide et plus stable. Cela pour stabiliser les mouvements de la partie mobile et rendre moins encombrant le scanner. Avec l'opportunité offerte par le CERISIC d'utiliser leur imprimante 3D, récemment acquise, l'ensemble des éléments structurels ont été réalisés grâce à cette imprimante. La modélisation des pièces a été réalisée sur le logiciel Autodesk Fusion 360.

Structure générale

Le principe choisi est de créer des petites pièces modulaires qui serviront à créer la structure au moyen de tiges filetées M8 en acier. Ces tiges donneront à la structure une certaine rigidité et stabilité en triangulant les surfaces carrées. Imprimer en 3D des petites pièces complexes pour agencer des grands éléments simples et facilement trouvables est intéressant sur plusieurs niveaux : cela permet d'économiser du temps d'impression et du filament tout en permettant de faire une structure de grande taille, et cela permet également de profiter des propriétés de l'acier pour les grands éléments structurants. Sur les figures suivantes, nous pouvons voir un aperçu de la version précédente et un rendu de la version actuelle. Certaines parties intéressantes de la structure sont également détaillées.

Figure 1: version précédente

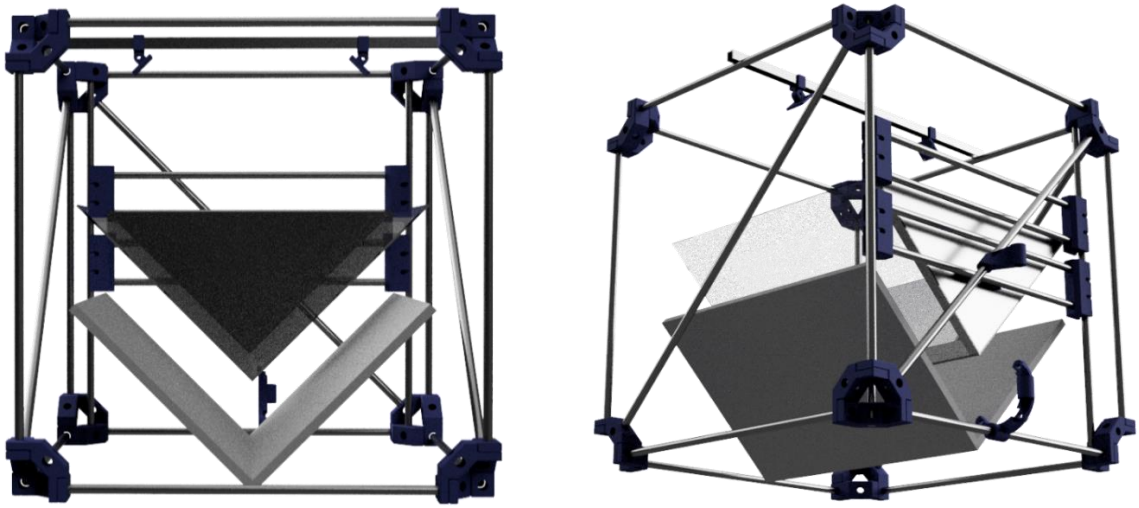


Figure 2 : rendus de la version actuelle

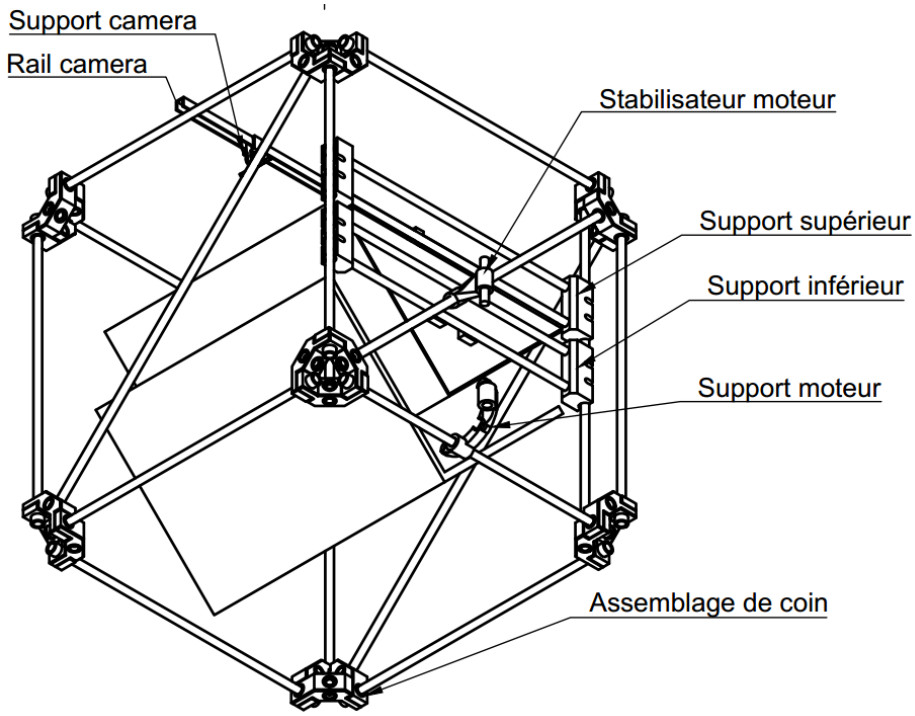


Figure 3 : éléments de la structure

Module de coin

Ce module a été conçu pour être petit et simple à imprimer, tout en permettant de le combiner pour créer un assemblage en 3D permettant d'y fixer les tiges filetées M8 structurelle. Plusieurs logements sont prévus pour fixer des tiges à angle droit, mais aussi à 45° afin de pouvoir trianguler la structure.

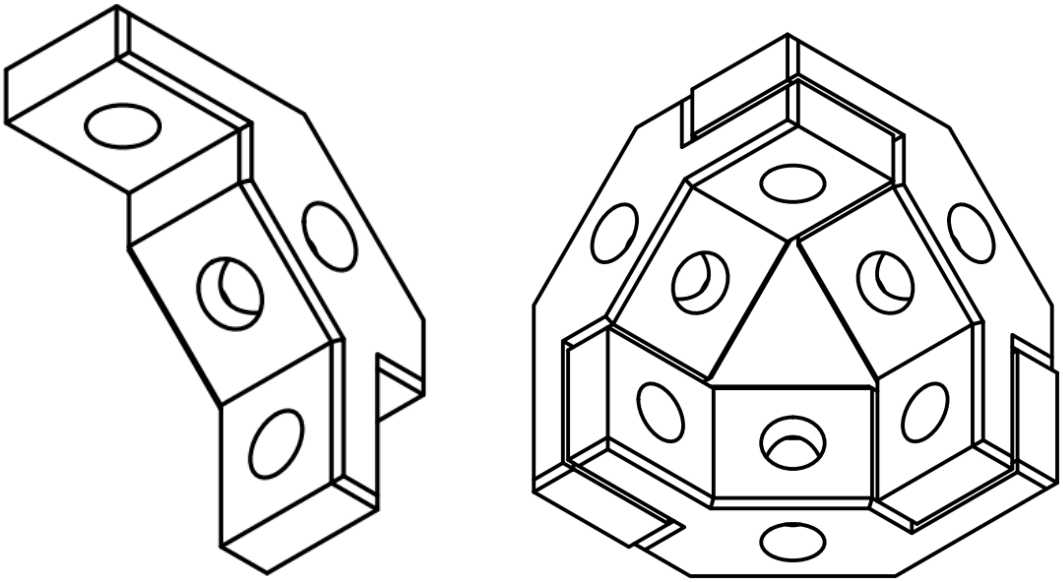


Figure 4 : module de base et assemblage en coin

Mécanisme de fin de course

Le mécanisme de la figure suivante permet de détecter lorsque la partie mobile presse contre le livre. En effet, la partie mobile est couplée au support inférieur, tandis que la courroie motrice est fixée au support supérieur. Lorsque le livre bloque la descente de la partie mobile, la motorisation continue de faire descendre le support supérieur jusqu'à l'enclenchement du switch de fin de course.

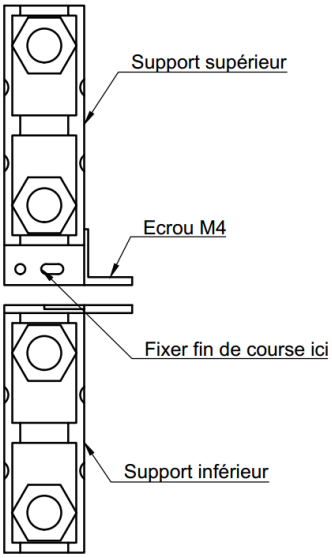


Figure 5 : mécanisme de fin de course bas

Support de caméra

Un petit mécanisme a été conçu pour fixer et régler sommairement les deux caméras sur leur rail. Il est illustré dans la figure suivante.

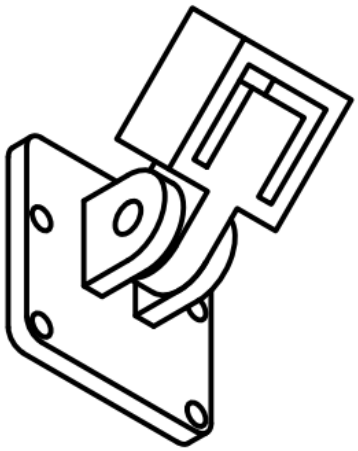


Figure 6 : Support de caméra

Partie électronique

Circuit multiplexeur de caméra CSI

Sur le précédent prototype, une caméra était utilisée pour prendre une photo de l'ensemble du livre et donc de deux pages. Sur le nouveau prototype, nous voulions prendre une image de chaque page séparément. Deux systèmes différents ont été considérés. Le premier consiste en l'utilisation d'une seule caméra pouvant se déplacer et pivoter pour prendre les photos des deux pages consécutives. La seconde option était d'utiliser deux caméras, soit une caméra dédiée par page.

Il faut donc relier deux Pi camera au Raspberry Pi. Ces caméras disposent d'une interface CSI. Cependant, le Raspberry Pi n'ayant qu'une seule entrée CSI, un circuit de multiplexage est nécessaire. Un tel circuit est disponible à la vente mais est relativement couteux (50 euros). La décision a donc été prise de concevoir un circuit et de le réaliser.

La Pi Camera utilise, pour envoyer ses données au Raspberry Pi, une interface CSI-2 (Camera Serial Interface – 2 lanes). C'est un type d'interface MIPI (Mobile Industry Processor Interface) qui utilise une ligne différentielle d'horloge et 2 lignes différentielles pour échanger des données en série à un débit qui peut être entre 100Mbps et 1Gbps. De plus, le Raspberry Pi utilise également une interface I²C pour configurer la caméra. Ce type de connexion est illustré sur la Figure 7.

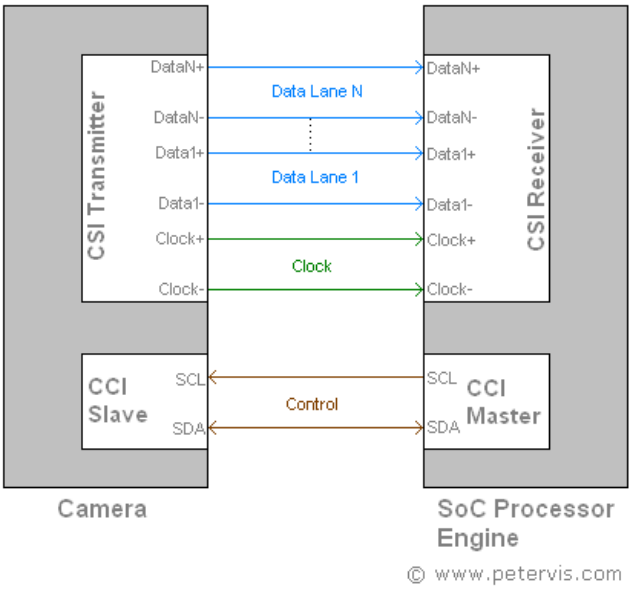


Figure 7 : interface CSI-2

Cela pose des contraintes sur le circuit à concevoir. En effet la haute fréquence des signaux séries sur les lignes différentielles ne permet pas d'utiliser des circuits multiplexeur classiques. Le circuit intégré FSA642, cependant, est prévu précisément pour gérer ce cas de figure. Pour commuter le bus I²C, beaucoup plus lent, un circuit intégré 74CBTLV3257 est utilisé (il s'agit d'un multiplexeur/démultiplexeur). Ce circuit permet de faire la commutation et permet de laisser passer des signaux bidirectionnels, ce qui est requis pour le bon fonctionnement du bus I²C. La Figure 8 montre le circuit qui a été conçu pour cette application. En plus des circuits intégrés, un pad est également exposé pour faire la sélection de la caméra à utiliser. La caméra A est celle sélectionnée par défaut grâce à une résistance de pull-down.

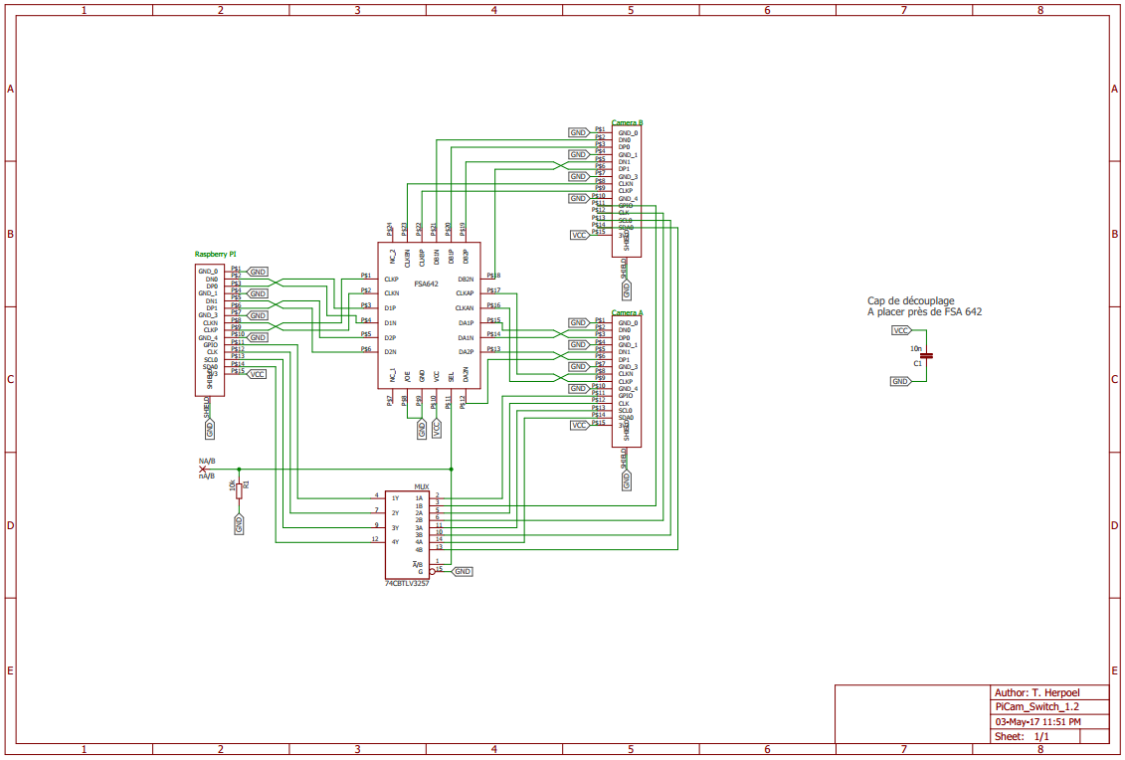


Figure 8 : Schéma du multiplexeur CSI-2

Liste des composants (BOM)

Nom	Package	Valeur	Quantité	Prix unitaire	Prix total
FSA642	QFN		1	0.76	0.76
74CBTLV3257	SOIC16		1	0.516	0.516
Molex 52271-1569	SMD		3	Échantillons	0
R1	1206	10K	1	0.01	0.01
C1	1206	10nF	1	0.10	0.10
PCB			1	1.5	1.5
TOTAL					2.886

Le prix de revient du prototype de cette carte est largement inférieur au prix de la solution vendue 50 euros et offre la même fonctionnalité.

Réalisation du PCB

La conception du PCB est réalisée au moyen du logiciel Eagle. La première étape est de créer les composants qui n'existaient pas dans les librairies de base, tels que la puce FSA642 et le connecteur Molex 15 broches qui permet de brancher les nappes des caméras.

L'utilisation de lignes différentielles à haute fréquence oblige d'être attentif à la longueur des pistes de chaque voie. En effet une longueur de piste différente entre le signal d'horloge et les signaux de données peut provoquer un décalage et corrompre les données.

De plus, ajuster l'impédance caractéristique de ces lignes devrait être adaptée correctement (100 Ohms entre piste positive et négative, 50 Ohms entre chaque piste et la masse). L'utilisation ici d'un PCB ayant 2

couches, et non 4 couches, ne permet pas facilement d'adapter ces impédances et cela n'a donc pas été fait. Cependant, étant donné que la nappe de connexion ne présente probablement pas cette impédance montre qu'adapter l'impédance n'est probablement pas critique pour le bon fonctionnement du système.

La première version du PCB a été fabriquée au moyen du site internet http://www.elecrow.com. La Figure 9 montre la version 1.2 du PCB, version légèrement retouchée du PCB qui a été fabriqué (visible en Figure 10. La différence entre les deux est la position du pad permettant de commander la direction du switch, proche, dans la nouvelle version, de la nappe allant vers le Raspberry Pi pour faciliter le câblage.

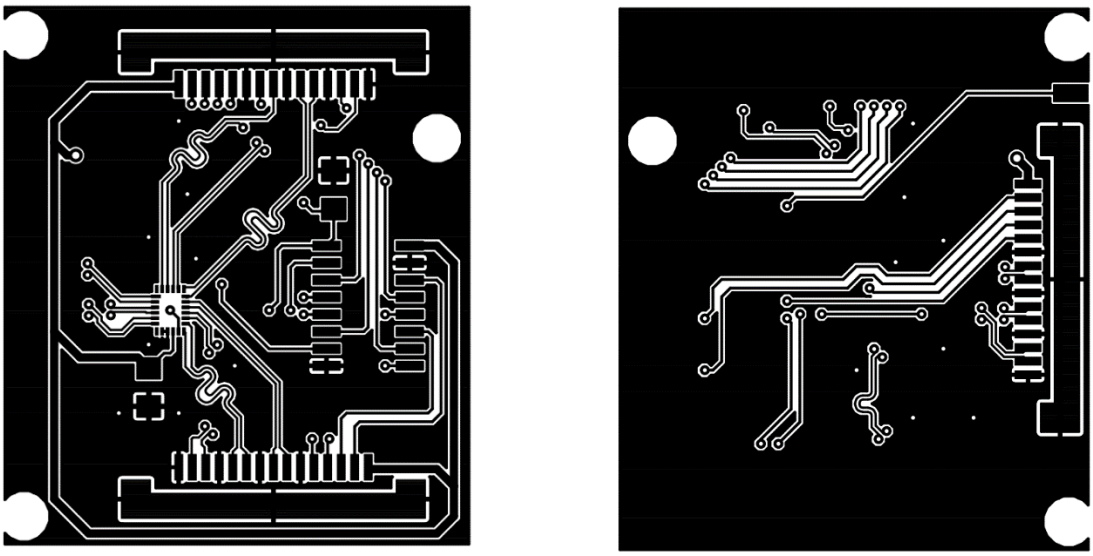


Figure 9 : Couches cuivre du PCB

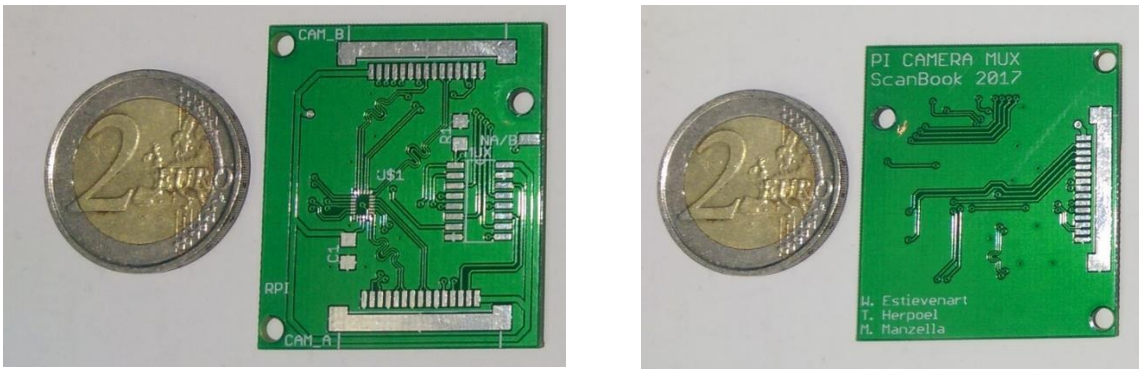


Figure 10 : PCB réalisé

Carte microcontrôleur

En plus du module de switch des caméras, il y a un PCB de contrôle de la partie mécanique. Celui-ci se compose d'un microcontrôleur et d'un driver pour le moteur pas à pas faisant bouger les vitres. Ce PCB développé l'année précédente a été repris et le programme du microcontrôleur amélioré. Ce PCB communique via le protocole USART avec le Raspberry PI. C'est ce dernier qui envoi les ordres de commande au microcontrôleur qui les interprète afin de piloter le driver qui a son tour commande le moteur. La première version de cette carte a été réalisée en 2015-2016 mais elle est adaptée cette année.

La broche /ENABLE du driver moteur est mise à la masse pour que le moteur soit alimenté dès que l'alimentation générale est présente. C'est la modification la plus important parce qu'elle permet d'éviter de détruire le driver du moteur lorsque l'alimentation du système est présente, mais que le moteur n'est plus alimenté. Si la partie mobile est en position haute à ce moment-là, il retombe sous l'effet de la gravité. L'énergie de ce mouvement crée une force électromotrice qui est injectée dans le driver moteur. Les diodes de roue libre intégrées dans la puce driver sont apparemment trop faible et la puce est détruite dans cette situation. Bloquer la broche /ENABLE à 0 permet d'avoir le module driver continuellement activé pour éviter la chute libre du mobile, tant que l'alimentation générale est présente.

Les interrupteurs qui activent le bus d'alimentation 5V et le bus d'alimentation prévu pour le moteur sont retirés. Un seul interrupteur est utilisé pour alimenter l'ensemble des systèmes.

Le régulateur 7805 n'est pas assez efficace pour alimenter le Raspberry Pi. Un convertisseur DC/DC équivalent, le OKI-78SR5, est proposé comme remplaçant. Il possède la même connectique et permet de fournir jusqu'à 1,5A à une tension de 5V facilement.

Le connecteur prévu pour communiquer avec un servomoteur est retiré. En effet, le système de caméra mobile a été abandonné au profit d'un système de deux caméras. Le servomoteur n'est donc plus utile.

Ces modifications sont réalisées sur le logiciel Eagle, mais la carte n'a pas été réalisée en pratique. L'ancienne carte a simplement été modifiée pour intégrer la protection du driver moteur du nouveau circuit.

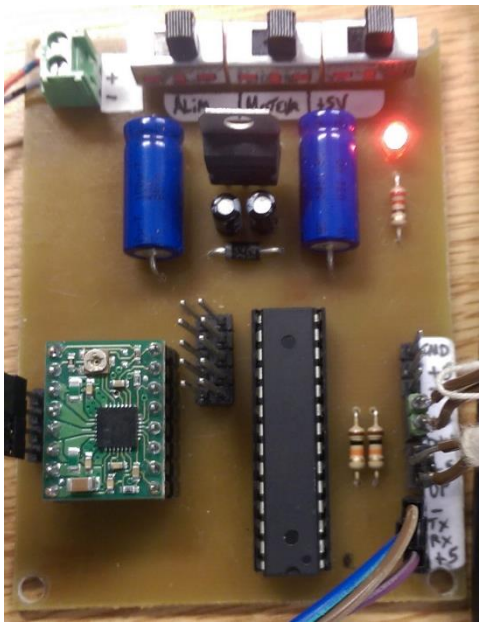


Figure 11 : ancienne carte modifiée

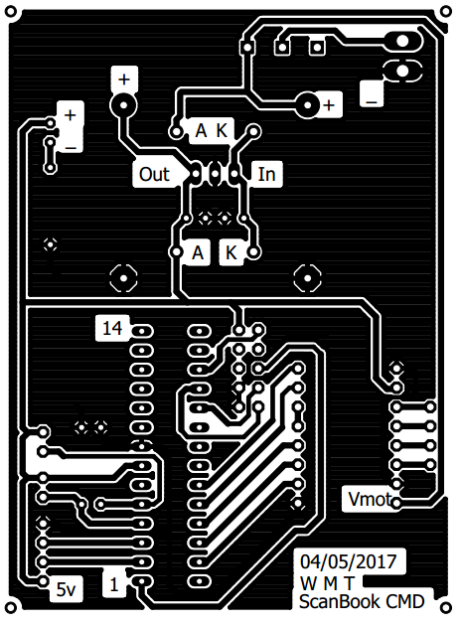


Figure 12 : nouvelle carte

Partie software

Programme principal

Le programme principal se trouve au sein du Raspberry Pi car comme dit précédemment, c'est lui qui envoi les ordres à la carte électronique de contrôle du moteur et c'est également lui qui effectue les photos et traitements d'images nécessaires à la construction du fichier PDF du livre scanné. Sur la Figure 13, on peut voir l'ordinogramme principal du programme contenu dans le Raspberry. Il y a tout d'abord une partie d'initialisation pour les variables, la communication USART et le moteur. Suite à cela se trouve trois grandes parties agissant ensemble pour former la séquence du scan. Le tout est imbriqué dans une boucle permettant de quitter le programme et d'arrêter le scanner si on le souhaite.

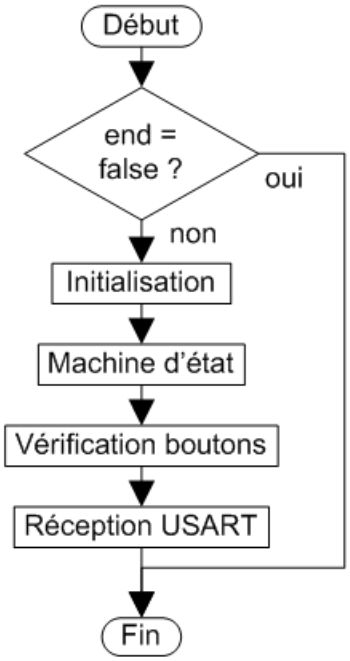


Figure 13 : ordinogramme principal

On peut voir sur la Figure 14 la machine d'état utilisée dans le programme. Selon l'appui sur les boutons, les traitements effectués et les messages reçu en USART, ses états vont évoluer et passer de l'un à l'autre afin de gérer la séquence du scan. Les numéros sur le graphique correspondent à des actions, que l'on retrouve dans le Tableau 1.

Tableau 1 : liste des actions

Numéro	Action
1	Bouton START
2	Rx USART => ISUP
3	Bouton NEXT
4	Rx USART => ISDOWN
5	Bouton END
6	Bouton STOP

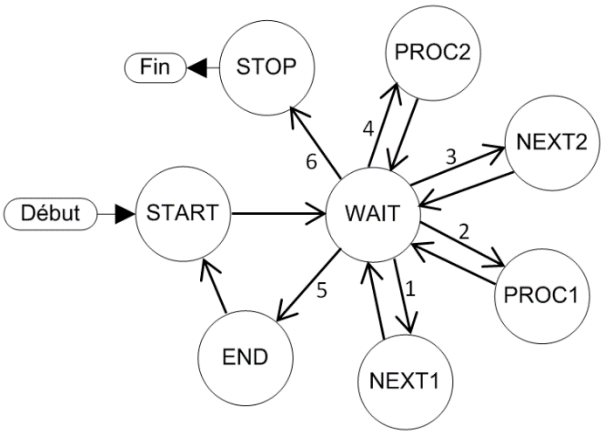


Figure 14 : machine d'états

L'ordinogramme sur le bloc « Machine d'état » se trouve à la Figure 15. On peut voir chaque série d'action associées aux états vu avant.

La séquence complète est donc la suivante (après l'initialisation) :

1. Afficher « appui sur START pour commencer un nouveau livre ».
2. Appuis sur START.
3. Envoi de la commande « UP » et verrouillage des boutons.
4. Réception de la commande « ISUP », afficher « appui sur NEXT pour continuer ou END pour finir » et déverrouillage des boutons.
5. Si appuis sur NEXT :
   - Envoi de la commande « DOWN » et verrouillage des boutons.
   - Réception de la commande « ISDOWN », affichage « scan en cours », traitement images, retour au point 3.
6. Si appuis sur END :
   - Afficher « livre fini », traitement PDF et retour au point 1.

À tout moment (lorsque les boutons ne sont pas verrouillés) on peut appuyer sur le bouton STOP afin d'arrêter le scanner et pour quitter le programme.

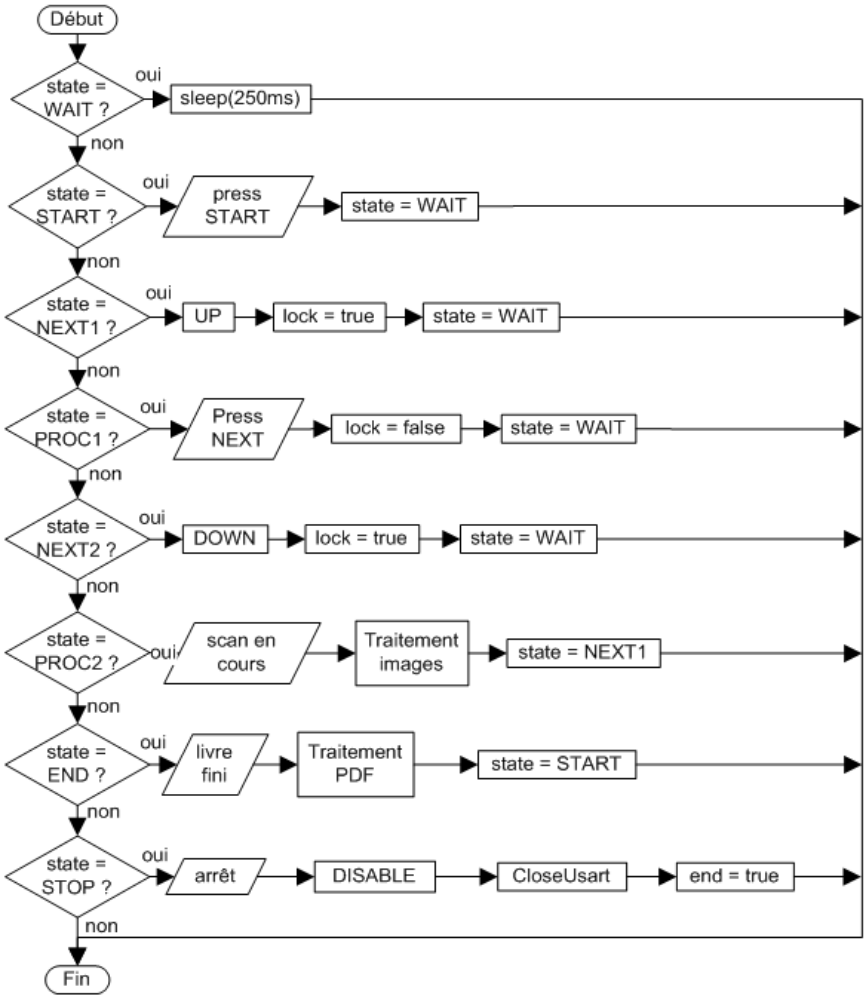


Figure 15 : test des états

On a vu dans la séquence complète que les boutons étaient verrouillés par moment. Cela est fait pour éviter des appuis intempestifs durant la procédure afin de ne pas générer un changement d'état alors qu'il y en a déjà un en cours (lors d'un mouvement des vitres par exemple). Sur la Figure 16, on peut observer l'ordinogramme correspondant au bloc « vérifications boutons » de l'ordinogramme principal.

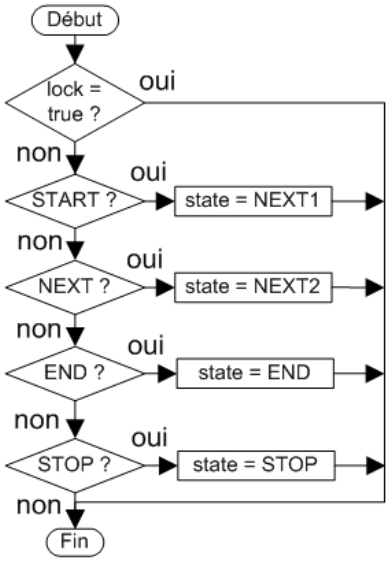


Figure 16 : test boutons

On va venir tester si un bouton a été appuyé et donc changer la variable d'état pour aller à l'état correspondant. Mais on teste les boutons uniquement si la variable de verrouillage n'est pas activée.

On peut voir sur la Figure 17le détail du troisième bloc de l'ordinogramme principal « Réception USART ». La fonction de réception USART fonction dans un thread en parallèle de la boucle du programme principal. Lorsqu'un message est reçu, une variable « dataAvailable » est activée afin d'aller traiter le message dans le programme principal. Dans le cas présent, on va vérifier si le message reçu est soit « ISUP » (quand les vitres son arrivée au fin de course haut), soit « ISDOWN » (pour le fin de course bas). Selon le message, on va envoyer une commande de désactivation au moteur par sécurité, on affiche le message correspondant à la position des vitres et on va mettre à jour la variable d'état pour évoluer dans la séquence.

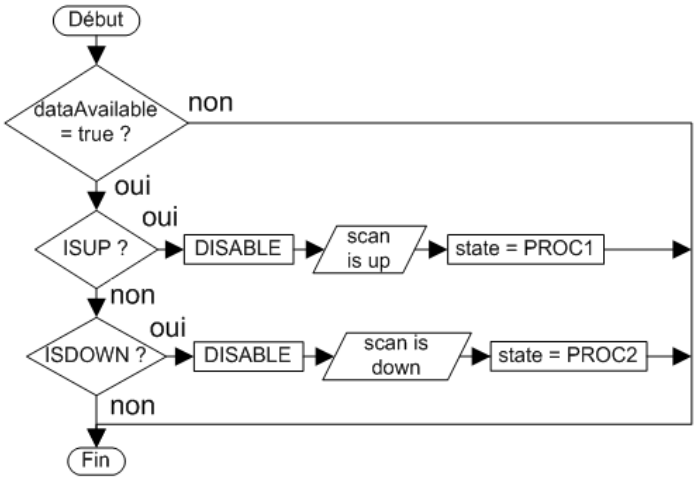


Figure 17 : réception USART

Dans l'ordinogramme de la machine d'état, on a pu voir un bloc de traitement image apparaitre. Ce bloc englobe toutes les étapes qui permettent le traitement des pages, de la prise de la photo à l'ajout dans le fichier PDF. Sur la Figure 18, on peut voir l'ordinogramme correspondant.

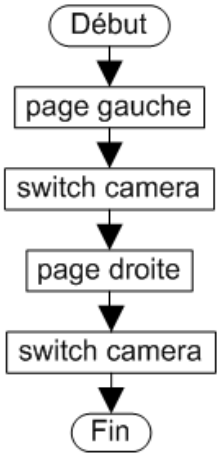


Figure 18 : traitement image

On voit qu'entre chaque traitement de pages, on utilise le switch caméra afin de changer de caméra pour la prise de vue. Et ce y compris avoir pris la deuxième photo, car il faut remettre le switch pour utiliser la caméra de la page gauche à la prochaine itération. Les blocs « page gauche » et « page droite » sont décrits en détails sur le Figure 19. Le changement entre les deux réside dans la rotation de la page. Car la photo prise n'est pas droite mais tournée à 90°. Et selon la caméra utilisée (et donc la page prise en photo), la rotation se fera dans le sens-horlogique ou anti-horlogique.

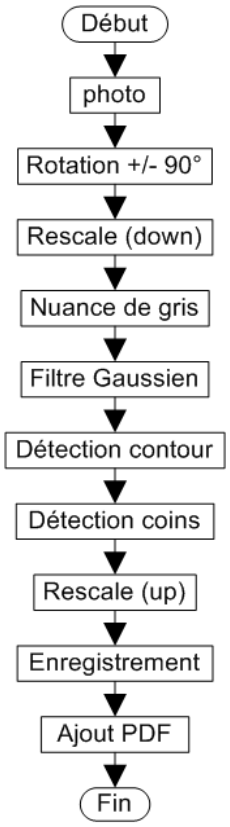
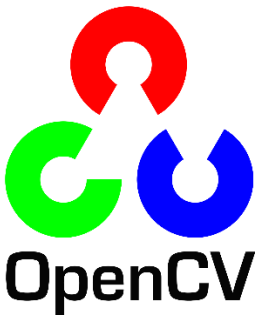


Figure 19 : traitement page

Traitement de l'image

En ce qui concerne le traitement des images, nous avons choisi d'utiliser la bibliothèque OpenCV et l'ensemble des opérations doit être effectué sur le Raspberry Pi. L'installation de la bibliothèque n'a pas été évidente car la compilation de celle-ci prend beaucoup de temps (pour une Raspberry Pi modèle B+, un peu plus de 24 heures). Après avoir installé la bibliothèque, nous avons commencé à faire plusieurs tests pour le traitement des images.



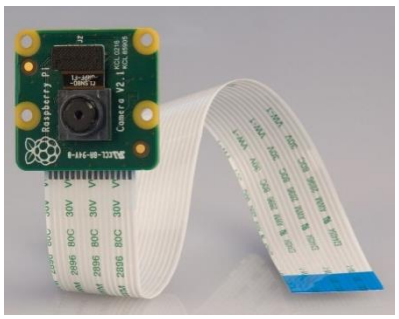


Figure 20 : Caméra Pi V2 8 MP

Tout d'abord, nous avons testé les caméras à notre disposition. Nous avons pu récupérer une caméra Pi version 1.3 de 5 MP d'un précédent projet car elle n'était plus utilisée. Mais comme nous voulions utiliser deux caméras dans notre prototype, nous avons commandé une seconde caméra. Cette dernière est un modèle supérieur à celle récupérée, la caméra Pi version 2.1 de 8MP. Nous avons commandé une caméra différente dans le but de pouvoir effectuer des comparaisons au niveau du traitement des images. Il s'est avéré que l'utilisation du switch caméra (présenté au point précédent) et de la configuration du Raspberry Pi au démarrage de ce dernier, ne permet pas l'utilisation de deux caméras de modèle différent. Nous avons alors utilisé une seconde caméra Pi version 2.1 d'un des membres du groupe.

Détection de contours

Au départ, le but était de pouvoir détecter le contour d'une image. Pour réaliser ce dernier, plusieurs étapes sont nécessaires.

Premièrement, l'image est transformée en niveau de gris. Cette opération permet de définir l'image par des nuances allant du blanc au noir. C'est une opération importante dans la détection du contour.

Deuxièmement, l'image est légèrement floutée par un filtre "Gaussian Blur". Cette étape permet de diminuer le niveau de détail de la photo et permet de facilité les calculs.

Troisièmement, une première détection de contour est réalisée par un filtre de Canny. A partir d'un certain seuil, il va déterminer les contours dans l'image. Un remplissage de certain contour est effectué par l'algorithme.

Finalement, on applique l'algorithme de détection de contour qui va récupérer les valeurs du contour des différents objets ou formes sur l'image et de les enregistrer dans un tableau. Un dernier algorithme permet de détecter le contour le plus grand et de dessiner ce dernier dans la couleur de notre choix.

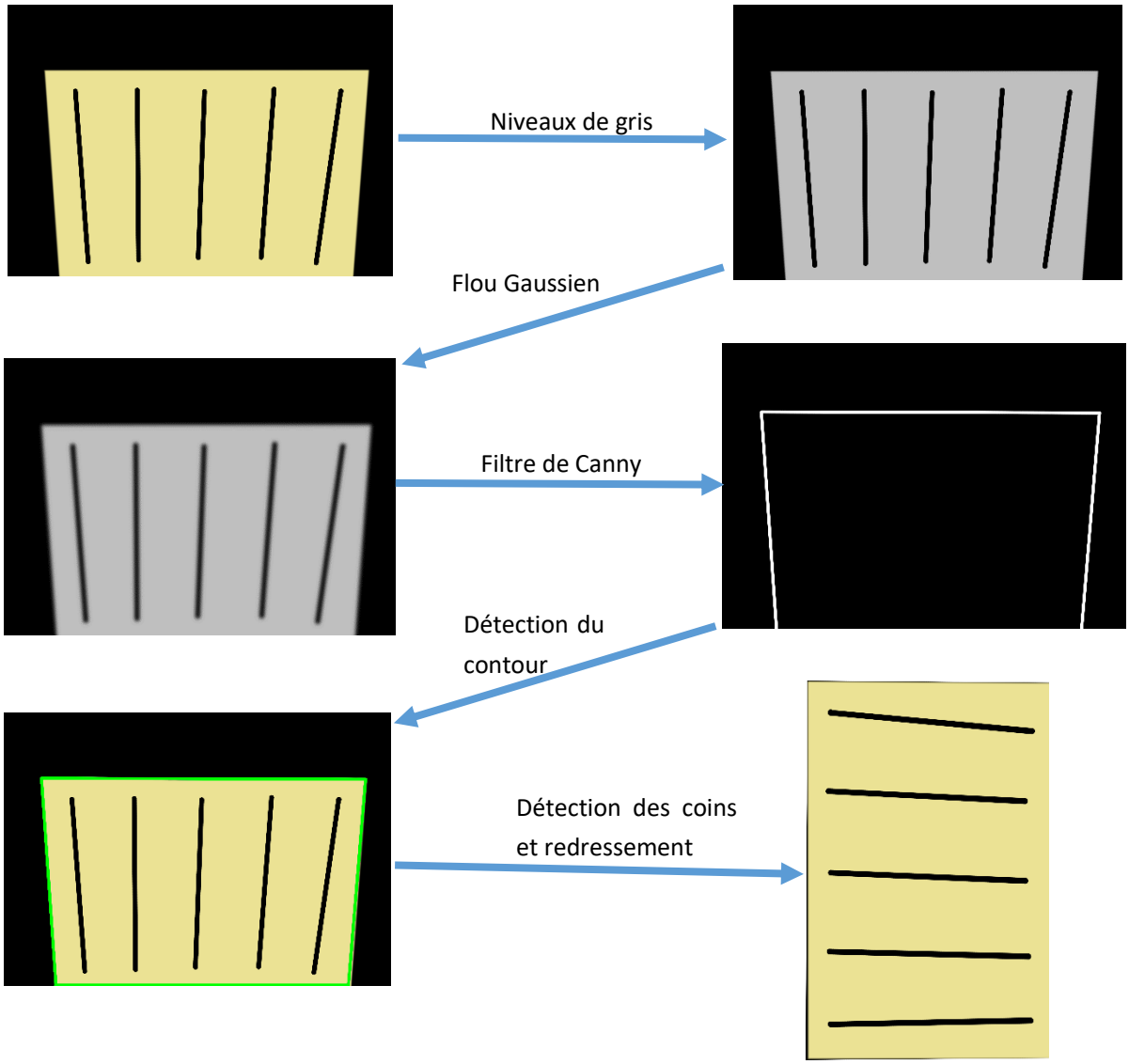
Redressement d'une page

On souhaite récupérer le contour d'une seule page. Cette opération est possible en récupérant les coordonnées des quatre coins de celle-ci. Ces coordonnées permettent par la suite de modifier l'image en l'étirant.

L'opération à réaliser est de déterminer le rectangle le plus grand et de dessiner par-dessus un nouveau rectangle dont on récupère les coordonnées des quatre coins.

Finalement, on peut étirer l'image en utilisant les coordonnées du rectangle et réadapter l'image par l'algorithme " warpPerspective".

La procédure simplifiée est illustrée sur les figures suivantes



Résultats et discussion

Mécanique

Des photos du prototype sont présentées en Figure 21 et Figure 22 . Elle montre que la réalisation suit effectivement le système conçu au moyen du logiciel de CAD. Les derniers éléments du prototype ont été assemblés avec les moyens du bord, par manque de temps pour dessiner des pièces et les imprimer. Cependant, le système modulaire prévu rempli correctement son rôle, et offre une base solide sur laquelle il est facile de monter des pièces supplémentaire (en les ajoutant sur les tiges filetées structurelles).

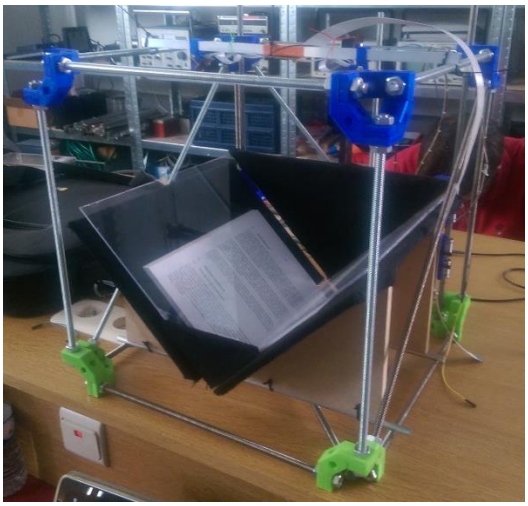


Figure 21 : prototype final

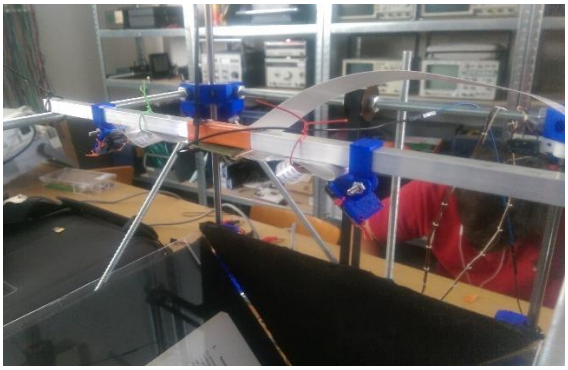


Figure 22 : montage des caméras

Un effort devrait donc encore être fait pour fixer de façon plus stable le mobile au support mobile inférieur, et pour fixer correctement la courroie au support mobile inférieur. De plus, il faut une fixation correcte pour le rail sur lequel se montent les caméras. Enfin, un support de montage de la carte multiplexeur CSI devrait être conçu.

Pour en finir avec la mécanique, fermer le système est primordial pour améliorer le contrôle de l'éclairage. Cette structure cubique permettrait de le faire facilement, en fixant des panneaux opaques sur les faces du cube.

Comparaison au cahier des charge

- Les caméras sont fixées au sommet de la structure.
  - Réalisé
- La structure est modulable.
  - Réalisé
- La structure est potentiellement fermée.
  - Ce point n'est pas encore réalisé, mais peut l'être facilement grâce à la forme du prototype.
- Réutiliser le support de livre de l'année précédente pour ce prototype.
  - Réalisé

Electronique

Multiplexeur de caméra CSI-2

Le circuit réalisé est illustré en Figure 23. Il a été testé indépendamment du système en vérifiant la continuité des entrées/sorties en fonction de la position du switch sélectionnée. Ensuite le circuit a été testé sur le Raspberry Pi, avec des caméras branchées sur les deux entrées du multiplexeur. Un petit script python a été réalisé pour faire basculer l'état d'une broche GPIO du Raspberry, connectée au pad NA/B du circuit, pour faire basculer l'entrée voulue.

Il en ressort que ce circuit fonctionne, mais uniquement si les deux caméras connectées sont les mêmes. En effet, brancher une caméra Pi 2.1 et une caméra Pi 1.3 ne fonctionne pas. L'explication la plus probable est qu'une demande d'identification du modèle de la caméra se fait au démarrage du Raspberry Pi à travers le bus I^2C. Le Raspberry se configure donc pour accepter les signaux de ce modèle de caméra uniquement.

Malheureusement, il est très difficile de vérifier cette hypothèse. En effet, cette partie du système Raspberry fait partie d'un morceau de programme propriétaire du GPU du micro-ordinateur. Il est donc impossible de vérifier le code source ou même de trouver de la documentation pour éventuellement reconfigurer le Raspberry Pi à chaque changement d'entrée. Cependant, le but du projet étant d'utiliser deux caméra PI 2.1, le circuit est fonctionnel et peut être immédiatement utilisé, à condition de se procurer une caméra Pi 2.1 de plus.

Figure 23 : circuit multiplexeur CSI

Carte microcontrôleur

Aucun driver moteur n'a été détruit depuis la modification effectuée sur la carte. La conclusion à en tirer est que cette modification a été efficace pour régler le problème. Le programme du microcontrôleur est simplifié et permet de communiquer avec le Raspberry Pi.

Comparaison au cahier des charge

- Concevoir un circuit électronique pour connecter deux Pi camera 2.1 au Raspberry Pi 3 afin d'effectuer les deux prises de vue (une pour chaque page).
    - Réalisé

- Réutiliser la platine électronique de 2015-2016 mais la modifier pour s'assurer de son bon fonctionnement.
    - Réalisé
- La gestion de la motorisation sera effectuée par un microcontrôleur relié au Raspberry Pi par un port série.
    - Réalisé

Software

Communication USART

La communication entre le Raspberry Pi 3 et le microcontrolleur ATmega88 se fait via Usart. Il a fallu configurer l'USART sur le Raspberry dans un premier temps avant de pouvoir l'utiliser. Une fois cette étape franchie, les différentes commandes pour contrôler le Scanbook on été elaborées.

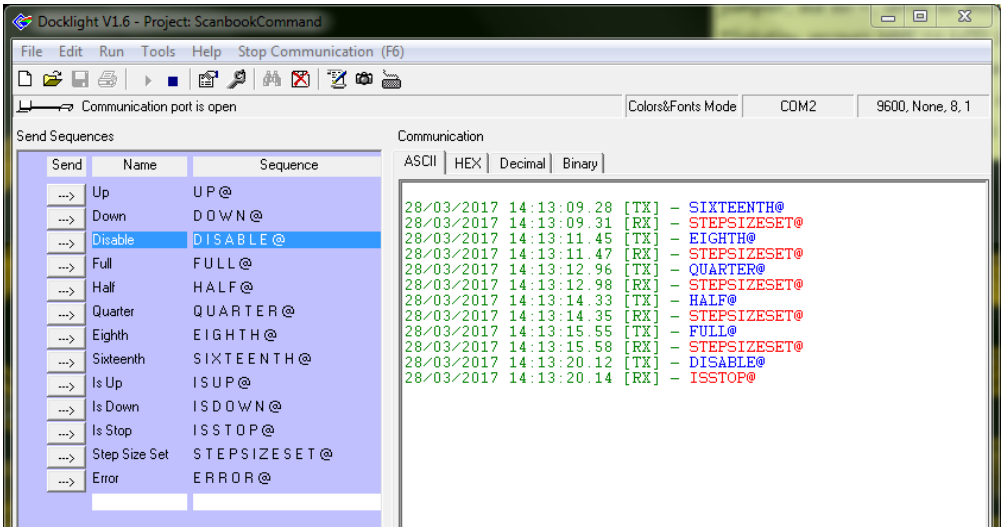


Figure 24 : commandes réglage résolution

Sur la Figure 24, on peut voir l'envoi de commande du pc (en bleu) avec le Docklight vers le microcontrôleur. Pour ce qui est du réglage de la résolution, le microcontrôleur répondra (en rouge) par la même chaine de caractère (cela dans le but de réduire et simplifier les commandes). On peut également voir une commande permettant l'arrêt du moteur si on le souhaite (pour l'arrêter pendant un déplacement par exemple, entre les deux fins de course) (voir Figure 25). On peut voir sur le panneau de gauche les différentes commandes configurées dans Docklight afin de réaliser tous les tests sur la communication USART. On peut aussi voir dans le bas, des réponses automatiques utilisées lors de test plus poussés sur les programmes pour simuler au mieux le fonctionnement réel de la communication entre le Raspberry et le microcontrôleur.

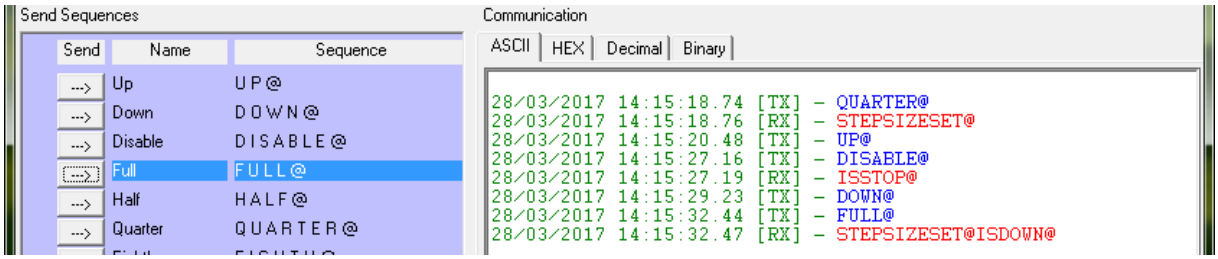


Figure 25 : Arrêt moteur et changement résolution pendant fonctionnement

On peut voir sur la Figure 25, un arrêt du moteur pendant la montée et un changement de la résolution pendant la descente.

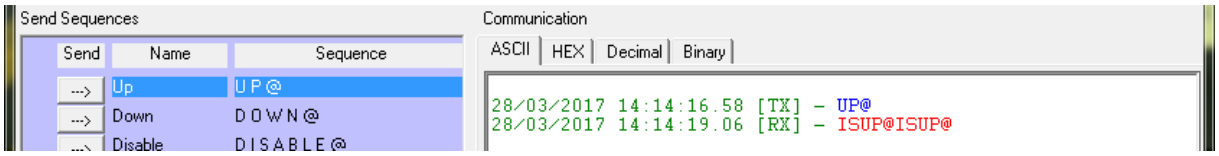


Figure 26 : commande montée vitres

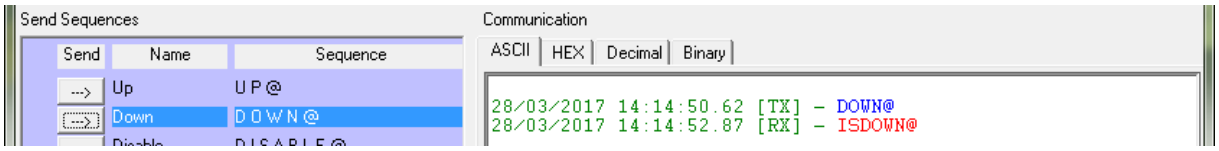


Figure 27 : commande descente vitres

On peut voir sur les Figure 26 et Figure 27, respectivement les commandes pour la montée et la descente des vitres envoyée (en bleu) vers le microcontrôleur. On voit la réponse du microcontrôleur (en rouge) lorsque la partie mobile actionne les fins de course haut et bas. Pour le fin de course haut, on peut remarquer que l'envoi s'est effectué deux fois. Cela est très certainement dû à un rebond du fin de course.

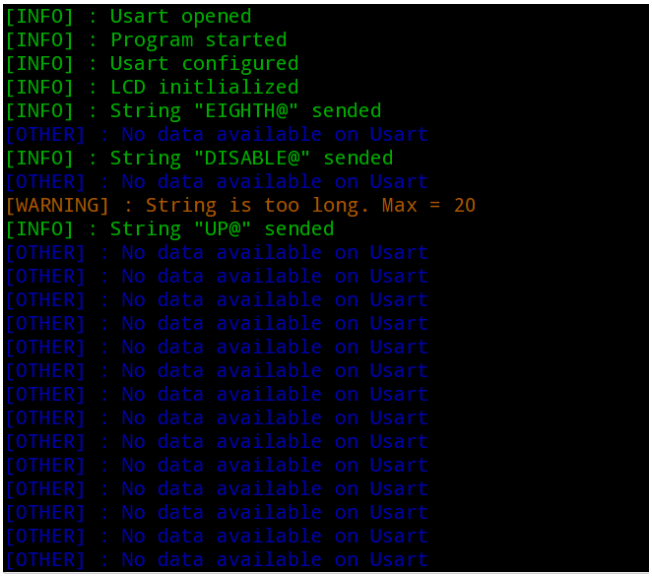


Figure 28 : Console Raspberry communication USART

Sur la Figure 28, on peut observer la console du Raspberry une fois le programme du Scanbook lancé. Un affichage couleur a été implémenté afin de visualiser plus clairement et rapidement les informations voulues. Il y a plusieurs type d'informations affichées et une couleur correspondant à chacune d'entre-elles. Les erreurs seront en rouges, les attentions en orange, les autres informations en vert ou en bleu selon leurs utilisations. On peut donc suivre le déroulement du programme et rapidement savoir où il en est. Sur cette capture d'écran, on peut voir une série d'initialisation comme l'USART ou le LCD (prévu en attente de la réception de l'écran tactile). On peut ensuite voir l'envoi en USART de deux commande, une pour régler la résolution et l'autre pour s'assurer que le moteur soi à l'arrêt. Vient ensuite un test d'envoi d'une chaine

trop longue, un message d'avertissement apparait en orange pour le signaler. Enfin, on voit que le Raspberry est en attente d'une réception USART.

```
[INFO] : Usart opened
[INFO] : Program started
[INFO] : Usart configured
[INFO] : LCD initlialized
[INFO] : String "EIGHTH@" sended
[INFO] : String "DISABLE@" sended
[OTHER] : No data available on Usart
[OTHER] : No data available on Usart
[WARNING] : String is too long. Max = 20
[INFO] : String "UP@" sended
[INFO] : String "ISUP" received
[INFO] : String "DOWN@" sended
[INFO] : String "ISDOWN" received
[INFO] : String "UP@" sended
[INFO] : String "ISUP" received
[INFO] : String "DOWN@" sended
[INFO] : String "ISDOWN" received
[INFO] : String "UP@" sended
[INFO] : String "ISUP" received
[INFO] : Program finished
[INFO] : String "DISABLE@" sended
[INFO] : Usart closed
```

Figure 29 : test commande scanbook

Sur la Figure 29, on peut voir un programme de test sur le Raspberry. Celui-ci consistait à faire monter la partie mobile, une fois en haut la faire attendre un certain temps, la faire redescendre et enfin attendre. La séquence tournait en boucle afin de simuler le fonctionnement final de l'application et de tester les programme en fonctionnement continu avec la structure physique. Sur la Figure 30, on peut voir la simulation de la partie microcontrôleur avec Docklight. On voit les commandes envoyées par le Raspberry (en rouge) et celle répondue (en bleu). Le programme a donc été testé en simulation avant de le tester sur la structure physique, afin de s'assurer qu'aucune erreur n'était commise.

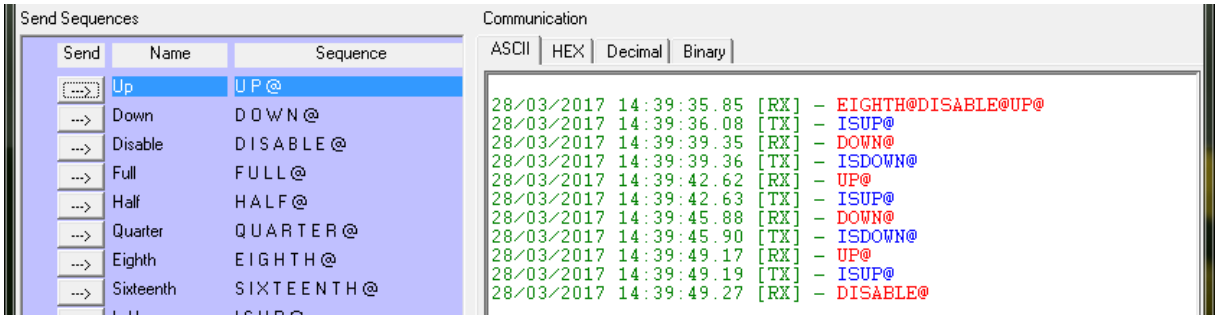


Figure 30 : Simulation microcontrôleur avec Docklight

Traitement de l'image

Comme expliqué dans le point 2.3 sur le traitement de l'image, il faut réaliser différentes opérations pour récupérer le contour d'une page.

Le premier test effectué était de récupérer le contour d'un livre avec ces deux pages (Figure 31). Tout d'abord, il faut appliquer sur la photo une transformation par nuance de gris. Il faut ensuite appliquer un filtre gaussien qui ajoute un effet de flou sur l'image ce qui donne comme résultat la Figure 32.

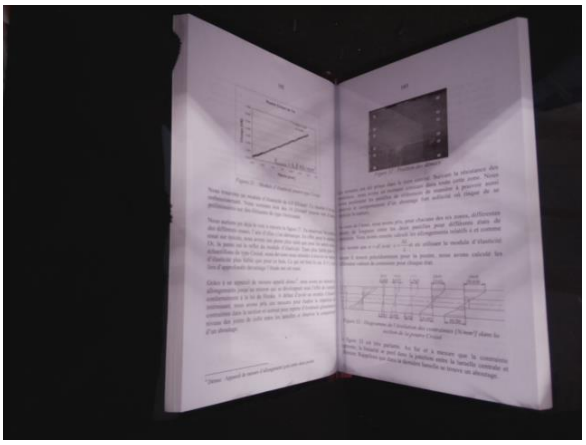


Figure 31 : Image originale (couleur)

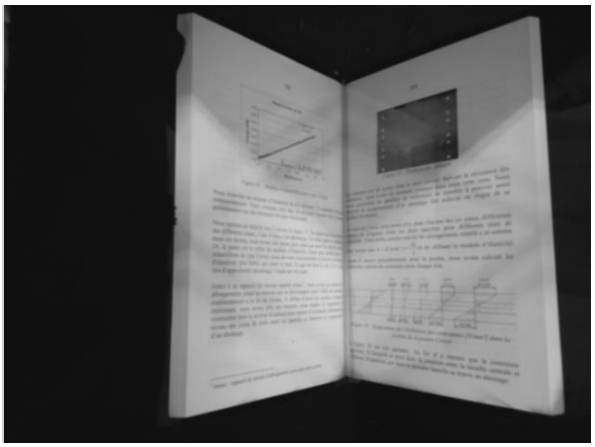


Figure 32 : Image transformée en nuance de gris et effet de flou

Remarque : Les photos utilisées pour les tests n'ont pas la résolution maximale de la caméra pour éviter des temps de calcul trop long.

Une première détection des contours est appliquée par un filtre de Canny. Avec ce premier traitement, nous obtenons un résultat qui se rapproche de ce que l'on recherche. Le résultat peut être affiné par le réglage d'un seuil (threshold).

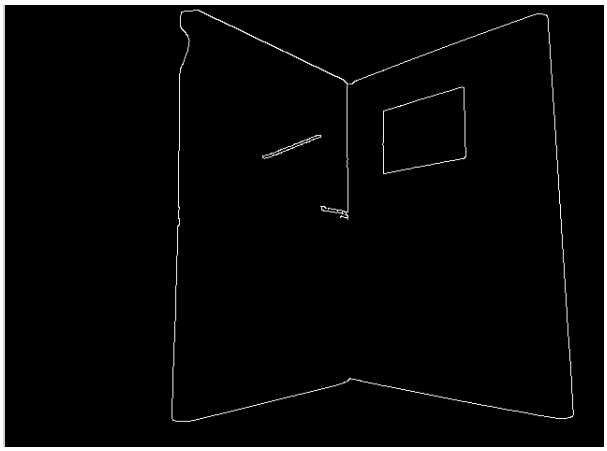


Figure 33 : Contour réalisé par un filtre de Canny

L'objectif suivant est d'obtenir des informations supplémentaires sur le contour en appliquant un algorithme qui a pour but de remplir certain espace. Cette étape permet de faciliter l'utilisation de l'algorithme suivant.

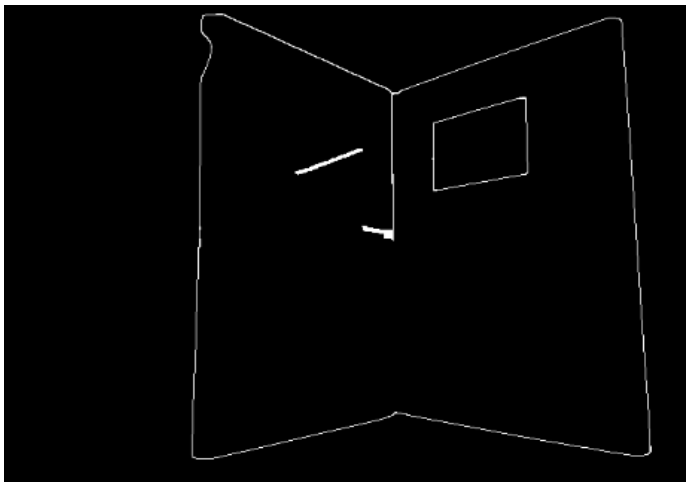


Figure 34 : Remplissage

Finalement, on applique l'algorithme de détection de contour qui va permettre de repérer sur l'image, les contours éventuels et de les enregistrer dans un tableau. Un autre algorithme permet de dessiner le contour sur l'image illustré en vert sur la figure suivante.

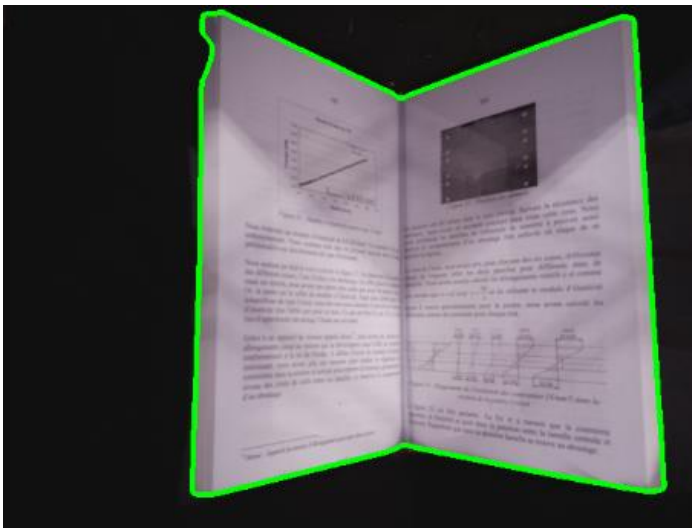


Figure 35 : Dessin du contour

Avec ce premier test, nous avons pu obtenir le contour de notre livre. Le prochain objectif était de pouvoir récupérer les quatre coins d'une seule page pour pouvoir réadapter l'image de celle-ci.

L'image récupérée par les caméras positionnées sur la structure nous donne une image par page mais la position de ceux-ci sont différentes en fonction de la caméra qui a pris la photo. L'image récupérée par la figure 27 a été récupéré par la caméra à droite de la structure alors que l'image à la figure 28 a été récupérée par celle de gauche. Il faut alors avant le traitement, effectuer une rotation de l'image de -90° si la photo a été prise par la caméra de droite, et de +90° si la photo a été prise par la caméra de gauche.

Figure 36 : Photos prise par la caméra droite

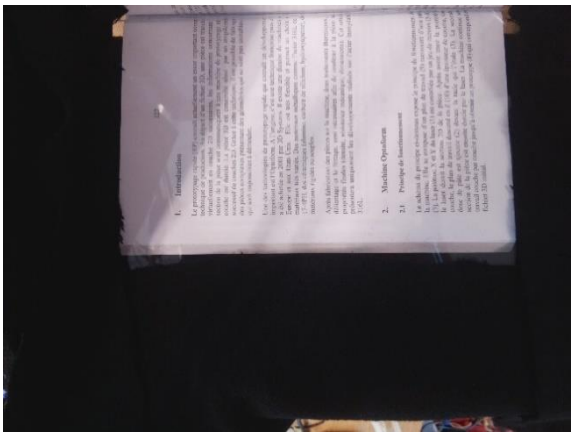


Figure 37 : Photo prise par la caméra gauche

Les opérations sont identiques qu'avec l'exemple précédent. Il faut faire une transformation en nuance de gris, puis appliquer la détection des contours. L'étape suivante a été de détecter les quatre coins du livre. Cette étape est effectuée car même si les caméras sont placées dans la meilleure position possible, elles verront toujours une partie de l'autre page et aussi ce qui se trouve sur le contour proche du livre. Il est donc important de pouvoir détecter uniquement la page dont on a besoin.

Figure 38 : Dessin du contour

La première opération est de détectant le rectangle le plus grand sur l'image et de l'entourer d'un nouveau rectangle dont on récupère les coordonnées qui permettent de réadapter l'image.

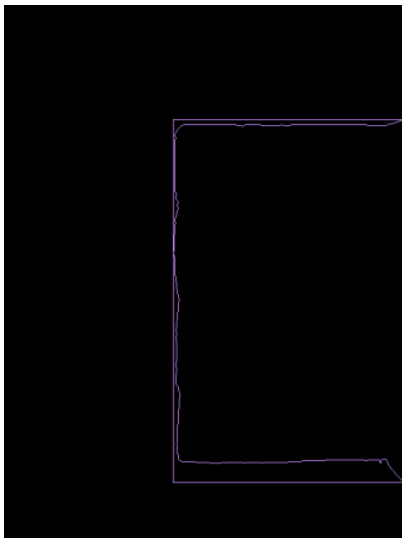


Figure 39 : Dessin d'un nouveau rectangle

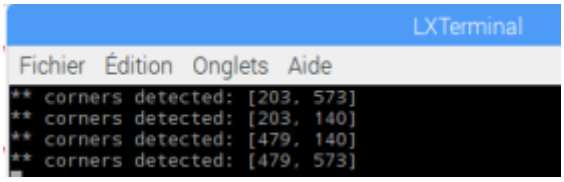


Figure 40 : Renvoi des valeurs des quatre coins

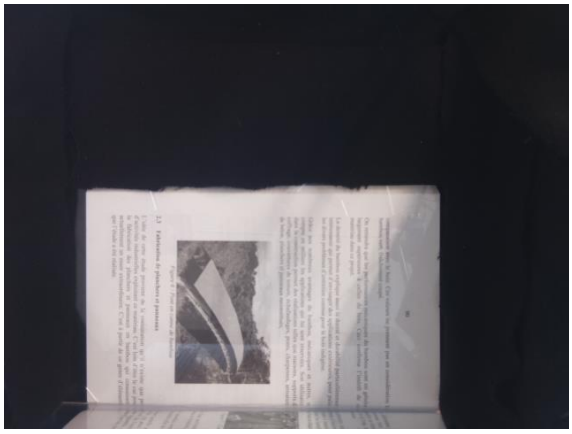


Figure 41 : Image originale

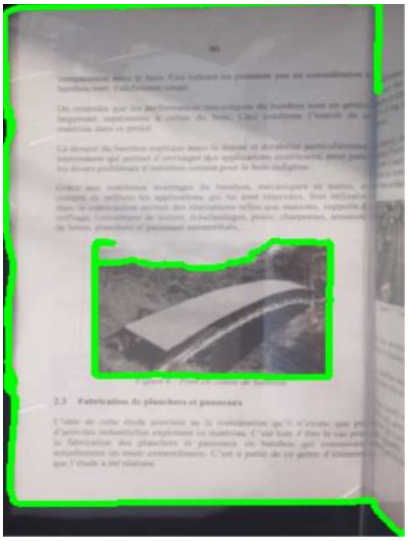


Figure 42 : Image redressée

Le résultat obtenu est satisfaisant mais l'image de la page n'est pas bien cadrée étant donné que l'algorithme détecte une partie de la seconde page dans son calcul de contour. D'autres problèmes persistent comme les reflets sur les deux vitres de plexiglass, ce qui fausse les résultats. Etant donné que l'on connait les zones "morte" du système, les images en dehors du champ d'actions, nous avons compensé cette zone en appliquant une bande noire. Cette option permet d'optimiser les résultats et d'éviter des calculs inutiles. Le résultat de cette manipulation est représenté par la figure suivante.

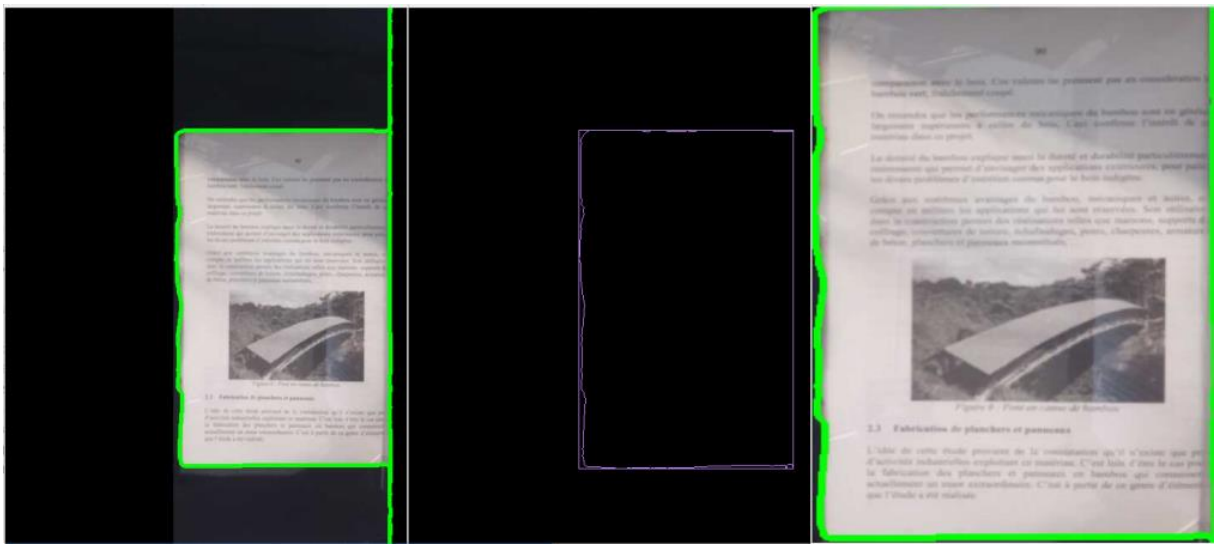


Figure 43 : Redressement de la page avec l'utilisation de bande noire

Le résultat obtenu par cette manipulation est beaucoup plus intéressant qu'avec la manipulation précédente. Cette solution n'est valable que si la taille du livre ne change pas et que la position des caméras est toujours la même.

Tous ces résultats constituent une idée du processus à réaliser pour atteindre la reconnaissance de pages voulue. Ils sont le fruit de plusieurs programmes de test et non d'un produit fini.

Comparaison au cahier des charge

- Utiliser OpenCV pour effectuer le traitement d'image et la reconnaissance de pages.
    - Le traitement de l'image est réalisé.
    - La reconnaissance de page a été travaillée. Les résultats ne sont pas parfaits mais sont encourageant. Lorsque les photos sont prises en haute résolution, la qualité d'image est plus que suffisant pour obtenir du contenu lisible.
- Création d'un fichier PDF avec les pages scannées.
    - Cette fonctionnalité n'a pas été réalisée mais des recherches ont été effectuées.
- Fonctionnalités : envoi du fichier PDF par mail, sauvegarde sur SD ou sauvegarde sur clé USB.
    - Cette fonctionnalité n'a pas été réalisée mais des recherches ont été effectuées.
- Coder en C++.
    - Réalisé.

Principaux problèmes rencontrés

Electronique : souder un composant QFN

Le circuit FSA642 est le seul à faire ce qui est voulu, à savoir multiplexer deux caméras CSI. Ce composant est principalement destiné au marché des téléphones mobiles ou tablettes, et n'existe donc que dans un package QFN très petit. Souder un tel composant a requis de travailler au microscope et au pistolet à air chaud. De plus, de nombreuses retouches de soudures, l'utilisation d'un décapeur thermique et probablement beaucoup de chance ont été nécessaires pour faire fonctionner ce circuit. Penser à un processus plus contrôlé tel qu'un four à refusions, ou un four à phase vapeur, pour réaliser de tels prototype pourrait être un grand plus.

Software : Installation de la bibliothèque OpenCV sur Raspberry

Avant de démarrer la conception du programme sur OpenCV, il a fallu installer la bibliothèque sur le Raspberry Pi. Au départ, nous n'avions pas de Raspberry Pi pour le projet. Nous avons donc utilisé nos Raspberry personnels pour effectuer les tests et installation avant l'arrivée des composants commandés.

Le problème était l'installation de la bibliothèque sur un Raspberry Pi B+. La compilation prend pour cet appareil, un peu plus de 24 heures. Ce qui nous a fait perdre beaucoup de temps.

Sur un Raspberry Pi 3, la compilation est plus rapide mais prend tout de même plusieurs heures.

Software : Communication Série

Pour faire communiquer le Raspberry et le PCB de contrôle, la communication USART devait être utilisée. Il existe se protocole sur le Raspberry mais il a été fastidieux à mettre en place pour une raison : l'implémentation de l'USART sur le Raspberry Pi 3 a été complétement modifiée par rapport à la version précédente (Raspberry Pi 2). Le nouveau modèle intégrant un Bluetooth embarqué, ce dernier utilise L'USART principal et c'est un USART secondaire qui est connecté aux broches d'entrées/sorties du Raspberry. Cet USART secondaire étant dépendant d'une horloge non fixe, nous ne pouvions pas l'utiliser pour une communication stable. Nous avons donc dû rechercher les configurations nécessaires afin d'inter changer les deux USART pour bénéficier du bon sur les GPIO.

Software : Configuration de l'écran tactile

Tout d'abord, il a fallu configurer le Raspberry pour lui permettre d'utiliser l'écran (utilisant le protocole SPI). Ensuite, il fallait s'intéresser à la réalisation d'une interface graphique affichable sur l'écran. Après discussion, nos recherches se sont tournées vers la librairie « Qt » (en C++) et l'outil de développement « QtCreator ». Ce dernier permet de créer une interface graphique rapidement avec son éditeur intégré. Une fois l'interface créée, il fallait l'ajouter au programme déjà développé en C++. Nous avons d'abord essayé d'intégrer au programme existant les fichiers de l'interface générés par Qt. Cela n'a pas été fructueux car le chargement des bibliothèques n'était pas bien configuré et une fois fait, la boucle de rafraichissement de l'interface prenait la main sur celle du programme principal.

Perspectives d'amélioration

Mécanique

1. Concevoir des pièces plus stables pour la fixation du mobile.
2. Concevoir des pièces pour la fixation de la courroie.
3. Concevoir des pièces pour la fixation du rail des caméras.
4. Concevoir une pièce de fixation pour le circuit multiplexeur CSI.
5. Concevoir un nouveau support pour le livre.
6. Fermer le système par des panneaux opaques. Cela permettrait de supprimer les reflets parasites sur les vitres de plexiglass, mais aussi de réduire l'influence de l'éclairage de la pièce sur le résultat visuel des prises de vue.

Electronique

1. Ajouter un système d'éclairage, combiné avec un système mécanique fermé, permettrait d'avoir un contrôle absolu sur l'éclairage des pages du livre. Un éclairage indirect de type LED pourrait être imaginé pour éclairer seulement la page étant photographiée. L'angle d'incidence de la lumière serait probablement important pour éviter les reflets sur la caméra.
2. Ajouter l'écran tactile pour réaliser l'interface homme-machine du scanner en y affichant une interface graphique pour le contrôler. Pour contrôler le scanner, on a pu voir dans la séquence et les différents ordinogrammes qu'il y avait 4 boutons utilisés : START, NEXT, END et STOP. Ces boutons ont été réalisés via des boutons physiques et les broches d'entrées sorties du Raspberry PI (GPIO). Cependant, nous avons voulu implémenter un écran tactile pour rendre le scanner plus esthétique et plus clair d'utilisation. Après quelque recherche, nous avons trouvé un écran compatible avec le Raspberry : 3.5inch RPi LCD (A), 320×480 Waveshare (contrôleur XPT2046). Comme dit plus haut dans les problèmes rencontrés, une tentative a été l'intégration des fichiers de l'interface dans le programme déjà réalisé. Cette tentative étant non fructueuse, une autre tentative imaginée serait d'intégrer le programme déjà réalisé du côté de l'interface (sous QtCreator).

Software

1. Création du fichier PDF. Il existe sur Linux une bibliothèque qui permet la conversion de fichier sous le format .JPG en PDF qui s'appelle « ImageMagick ». Il suffit par l'intermédiaire d'une commande de sélectionner les fichiers (images des pages) que l'on souhaite convertir et cela nous donne le fichier PDF en sortie.
2. L'envoi du fichier par mail peut être effectué par la bibliothèque « mpack » qui permet l'envoi d'une pièce jointe par mail à travers la console. Le Raspberry Pi 3 à la possibilité d'être connecté à un réseau par un câble Ethernet RJ45 ou par liaison Wifi.
3. Une autre stratégie pourrait être investiguée pour détecter la forme de la page. Il devrait être possible de poser un quadrillage de QR code sur le support, en dessous du livre. Cela permettrait à la librairie openCV de facilement reconnaitre les contours de page. Il suffirait de regarder à partir de ou les QR code ne sont plus visibles pour définir le contour du livre.
4. Une fois les pages correctement traitées et compilées dans un PDF, il pourrait être intéressant d'utiliser de l'OCR pour reconnaitre le contenu textuel de ces pages.

Conclusion

L'objectif de ce projet était de passer du premier prototype, qui servait à illustrer la preuve de fonctionnement du concept, à une version fonctionnelle sur laquelle le travail pouvait être continué dans l'espoir d'arriver à un produit final qui répond totalement à la demande.

Pour cela, la première étape a été de créer une nouvelle structure répondant aux attentes du cahier des charges. Cette structure est plus solide, plus stable et permet d'être adaptée et améliorée facilement.

Le traitement de l'image a été travaillé avec le même objectif. Le traitement effectué par le système précédent était également une preuve de concept réalisé dans un langage fermé et peu facile à améliorer. Cette nouvelle version a jeté les bases de ce traitement en utilisant un environnement de développement et des outils plus professionnels et performant. Le traitement permet donc, actuellement, de récupérer et de modifier l'image de manière à la rendre la plus nette possible. Il permet également de détecter grossièrement une page et de la transformer et redimensionner en une page droite et plate.

La qualité de prise de vue a également été considérablement augmentée. Le choix des deux caméras nous permet d'obtenir une résolution équivalente de 16MPixels, là où le prototype précédent n'offrait que 320.000 pixels. Pour que ce système soit fonctionnel, le projet doit encore se munir d'une caméra Pi 2.1.

Enfin, le caractère multidisciplinaire de ce projet nous a permis d'acquérir de l'expérience dans différents domaines, comme la conception de pièces en 3D, la conception de PCB, la configuration d'un système embarqué linux sur un Raspberry Pi et le traitement de l'image grâce à OpenCV.