TP caméra CCD

Programmation

Ce TP doit permettre à l'étudiant(e) de se familiariser avec la notion de **chronogramme**. Le chronogramme est le séquencement d'ordres logiques (signaux d'horloges) dans le temps pour actionner les phases de transfert d'un capteur CCD et de traitement de la chaîne électronique (cf. le cours). Selon le chronogramme employé, on peut avoir accès à différents modes de lecture de l'image et donc différentes "formes" de l'image numérique. Ce TP permettra également de se rendre compte du besoin de synchronisation entre le rasberrypi (via le programme de contrôle Code_Python.pdf) et la caméra pour assurer un bon fonctionnement de cette dernière (cf. *datasheet* du Kodak AF400).

Allumage du rasberrypi : Suivre les intructions données sur la page A4 plastifiée qui se trouve sur la paillasse. En cas de doute, merci de demander de l'aide à l'enseignant(e).

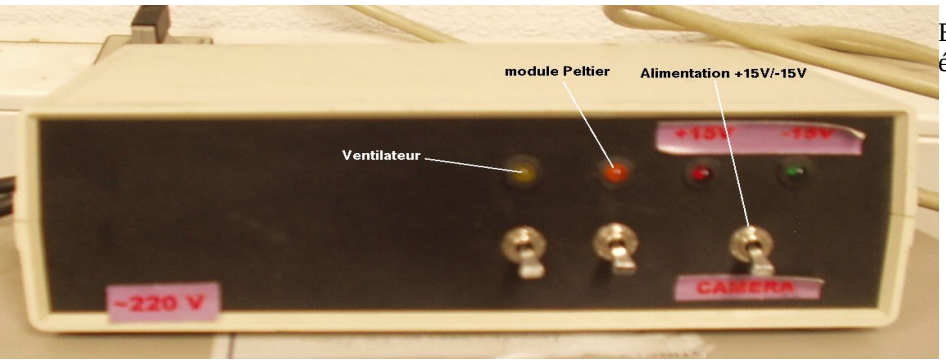
Allumage de la caméra :

1) Allumer l'interrupteur général à l'arrière du boîtier noir d'alimentation.

2) Mise sous tension : interrupteur +/-15 V. **Ne pas allumer le refroidissement par l'élément thermo-électrique Peltier** (interrupteur du milieu).

3) Mise en marche du ventilateur.

Pour éteindre la caméra, il suffit de reprendre les instructions ci-dessus en sens inverse.



Boîtier d'alimentation de la caméra Audine équipée du capteur CCD Kodak AF400.

1. Le programme d'acquisition

Le code main.cpp permettant de piloter la caméra est donné dans le fichier Code_Python.pdf.

Le code **main.cpp** et l'exécutable **main** se trouvent dans le répertoire **~/guitastro/gui/tp_audine_elec**.

Ouvrir un terminal sur le Bureau et taper la commande : **cdprog** => Cela vous placera directement dans le bon répertoire.

Pour lancer le programme main.cpp en ligne de commande, taper dans le terminal :

./main full_frame 1 2 nom_fichier_image.fits

(1 = 1 s de temps de pose ; 2 pour un binning 2x2)

N'oubliez pas d'adapter les paramètres d'entrée en fonction de vos cas d'utilisation.

2. Analyse du code main.cpp

Cette partie du TP est à préparer à la maison avant le TP. Vous rendrez à l'enseignant(e) un compte-rendu préliminaire incluant des éléments de réponse sur les questions ci-dessous en début de séance.

1) Prendre le temps de lire et de comprendre chaque ligne de la fonction **tp_acquisition_fullframe** du fichier Code_Python.pdf.

   Expliquer sur votre compte-rendu quelles sont les principales étapes realisées par cette fonction (utiliser les commentaires pour vous aider).

   A quoi correspondent ces différentes étapes physiquement pour la caméra ?

   Dans ce fichier, on utilise une structure (camprop) pour assembler toutes les constantes de la caméra. On notera en particulier l'emploi d'éléments suivants de cette structure :

   - **nb_photox** : nombre de cellules CCD actives sur l'axe X (=768)
   - **nb_photoy** : nombre de cellules CCD actives sur l'axe Y (=512)
   - **nb_deadbeginphotox** : nombre de cellules CCD masquées au début du registre horizontal (=14)
   - **nb_deadendphotox** : nombre de cellules CCD masquées à la fin du registre horizontal (=14)
   - **nb_deadbeginphotoy** : nombre de lignes masquées au début de la matrice (=4)
   - **binx** : valeur du binning X
   - **biny** : valeur du binning Y
   - **exptime** : temps de pause (en secondes)

2) En vous aidant du schéma du capteur CCD Kodak AF400 (Figure 1), associer les variables ci-dessous aux différentes parties du capteur.
3) De quel type de capteur CCD s'agit-il ? Combien y a-t-il d'électrodes par photosite ?
4) Décrire la manière dont le transfert de charges est effectué pour ce capteur. Combien d'horloges sont nécessaires pour assurer le transfert des charges ?

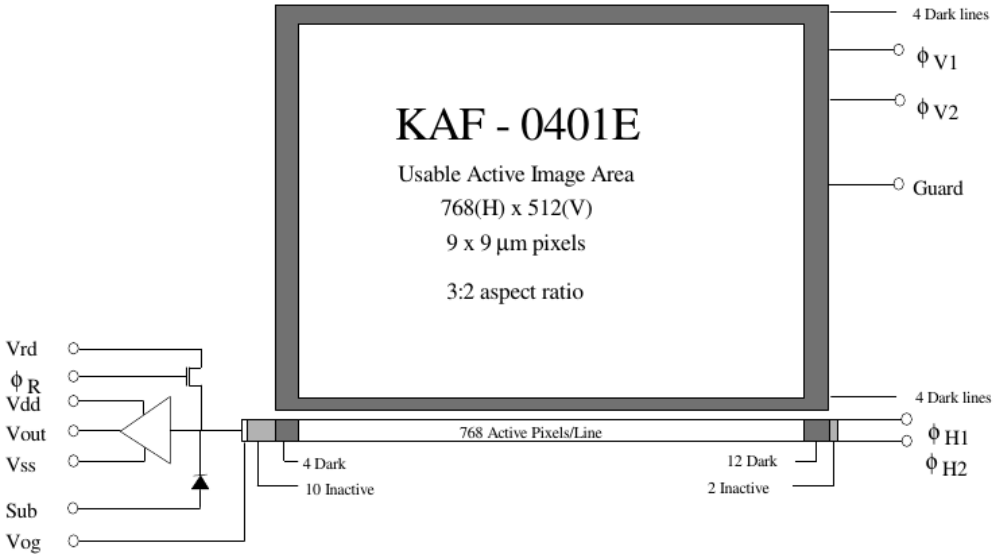


Figure 1 - Schéma du capteur CCD (cf. *datasheet* du CCD)

5) Analyser en détails les deux fonctions **tp_fast_vidage()** et **tp_read_frame()**.

Expliquer sur votre compte-rendu les différentes actions réalisées par ces deux fonctions.

Ne pas oublier d'expliquer le rôle des sous-fonctions incluses dans ces deux fonctions notamment **tp_zi_zh**, **tp_read_pel_fast** & **tp_read_pel_fast2**.

Pour chaque fonction, identifier les horloges actives et ce qu'elles permettent de contrôler sur le capteur et la chaîne électronique.

Remarque 1: La fonction **bcm2835_gpio_write()** est une fonction dont l'exécution permet de commander l'état des horloges du capteur et de la chaîne électronique. La fonction **tp_sleep(d)** permet alors de créer un palier d'une durée d.

Remarque 2: Bien lire les commentaires au début du programme qui identifie notamment la valeur binaire des états haut et bas.

Remarque 3 : A la fin du programme, vous trouverez une série de commande définissant la valeur par défaut de certaines horloges (cf. ci-dessous). Si l'état de l'horloge n'est pas explicitement modifié dans les autres fonctions, l'état de l'horloge est donné dans liste ci-dessous.

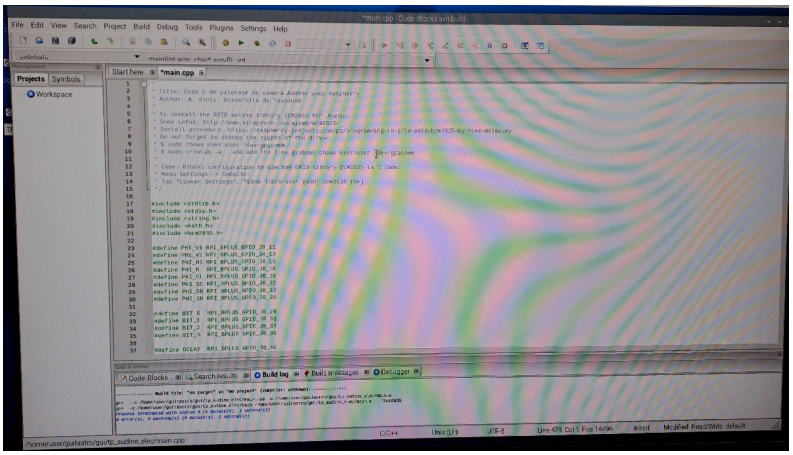
```
// -- Set default values for outputs
bcm2835_gpio_write (PHI_V1, LEVEL_LOW) ;
bcm2835_gpio_write (PHI_V2, LEVEL_LOW) ;
bcm2835_gpio_write (PHI_H1, LEVEL_HIGH) ;
bcm2835_gpio_write (PHI_R,  LEVEL_LOW) ;
bcm2835_gpio_write (PHI_CL, LEVEL_LOW) ;
bcm2835_gpio_write (PHI_SB, LEVEL_LOW) ;
bcm2835_gpio_write (PHI_SC, LEVEL_LOW) ;
bcm2835_gpio_write (PHI_SN, LEVEL_LOW) ;
```

6) Dessiner les chronogrammes des différentes sous-fonctions de la fonction **tp_fast_vidage** en respectant les durées temporelles de chaque palier.
7) En déduire les chronogrammes créés par la fonction **tp_fast_vidage**.
8) Construire les chronogrammes de la fonction **tp_read_frame**. En déduire comment le signal de sortie du CCD évolue au cours du temps. Vous pourrez vous aider du cours pour ce faire.
9) A quoi correspond le binning ? A quoi peut-il servir ?

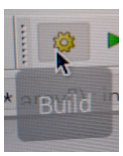
3. Visualisation & modification de main.cpp

Sur le Bureau, cliquer sur l'icône **TP Prog**, puis sur **Execute**.

L'interface Code Blocks va s'ouvrir montrant le programme main.cpp.
Les commentaires dans le code sont écrits en vert.



Après modification du code, appuyer sur l'icône **Build** pour compiler le code.



Vérifier que la compilation s'est bien passée dans l'onglet **Build log** (fenêtre en dessous du programme).

Utiliser les instructions de la Section 1 pour exécuter le programme en ligne de commande.

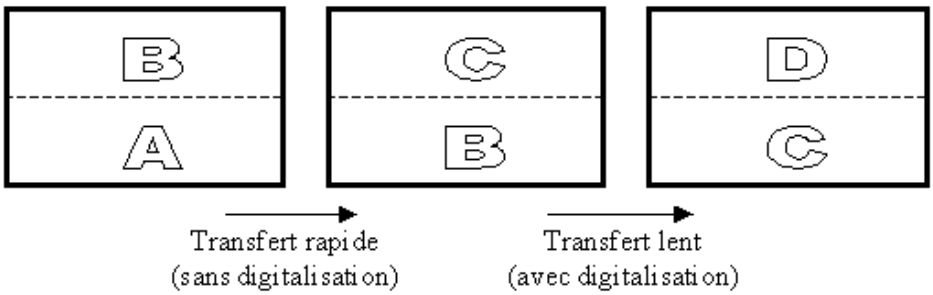
4. Création du mode demi-trame

4.1 Effet de smearing

1) Ouvrir l'interface Audela en cliquant sur **TP CCD** sur le Bureau, puis **Execute**. Une interface utilisateur va s'ouvrir. Le temps de pose et le binning peuvent être modifiés en haut de l'interface. Sélectionner **Method** = *full_frame*. Pour prendre une image cliquer sur **Acquisition**.
2) Faire une image du rond blanc sur fond noir sur un temps de pose de 1 s en binning 1x1. La caméra étant sensible à la lumière, la prise d'image se fera dans le noir avec la porte légèrement entrouverte.
3) Refaire la même image sur 5 s. Commenter la différence.
4) Refaire une image sur 5 s en mettant le cache en carton devant l'optique après les 5 s et ce jusqu'à l'apparition de l'image sur l'écran.
5) En déduire l'origine du smearing.

4.2 Fonction de lecture demi-trame

Nous désirons créer la fonction de lecture demi-trame. Ce mode d'acquisition consiste à vider sans numérisation les N/2 premières lignes (partie A), puis à lire normalement les N/2 dernières lignes (partie B). De cette façon, nous pouvons réduire considérablement l'effet de smearing d'un objet brillant sur fond noir.



Principe de lecture du mode demi trame. Seule la partie B sera numérisée*.*

Son transfert rapide vers la zone occupée précédemment par A permet d'éviter

le smearing. Ce mode est bien adapté si la zone A n'est pas éclairée !

1) Modifier la fonction **tp_acquisition_fullframe** pour l'adapter aux besoins. **Procéder pas à pas en notant toutes vos modifications pour mieux tracer l'origine des erreurs lors de la compilation du programme.**

2) Compiler le programme modifié en appuyant sur **Build** (cf. Section 3). Expliquer sur votre compte-rendu ce que vous avez modifié en le justifiant.

3) Faire l'image du rond blanc sur fond noir sur 1 s en binning 1x1 (attention de bien positionner le rond blanc !) et vérifier ensuite que l'image ne montre pas de smearing. Utiliser les commandes en ligne de commande de la Section 1.

4) Est-ce que cela fonctionne également lorsque le binning est changé ? Faire un essai en

binning 2 x 2.

5) Quelle autre technique pourrait être utilisée pour supprimer l'effet de smearing ? Justifier votre réponse.

6) Sauvegarder les images prises dans un répertoire à vos noms sur le Bureau ainsi qu'une copie du programme modifié à la fin de la séance de TP.
   Les images créées seront au format .fits et stockées dans le même répertoire que le programme main.cpp.