Université du Littoral – Côte d'Opale
Licence de Physique-Chimie et Physique – troisième année – Calais

Etude d'une réaction chimique oscillante

Réaction de *Belousov–Zhabotinsky*

Amaury Kasprowiak[b], préparation de la réaction (sec. 2) et mise au point, Dk
Pierre Kulinski[a], réalisateur mesures photométriques et vidéo (sec. 3), Dk
Sylvie Passepont[b], préparation TP chimie sur Calais,
Dmitrií Sadovskií [a*], initiateur du projet, développement informatique (sec. 4, A,B)
Anton Sokolov[a], intervenant TP, montage et enregistrement informatique initiaux
[a]*Département de physique*, [b]*Département de chimie*

Calais, automne 2020

1 Introduction

Dans sa 1re partie «vibrations», le cours magistral «Vibrations et ondes» du 1er semestre de L3 physique et L3 chimie–physique a pour son but d'apprendre les étudiants de ces deux formations de savoir reconnaître les processus oscillatoires dans des phénomènes divers et de les analyser selon les principes et les techniques communs pour tout tels phénomènes. C'est dans cet esprit que nous avons décidé d'élargir la gamme des systèmes oscillatoires étudiés en TP du 1er semestre. Nous pensons que l'origine chimique de ce phénomène motivera les étudiants, dont la plupart ont choisit la filière chimie–physique, à mieux réaliser et comprendre leur TP.

Remerciements

Nous tenons à remercier François Delattre (chimie, Dk), et Arnaud Cuisset (physique, Dk) pour leur soutien de ce projet. Nous remercions Emmanuel Vitoux (informatique, Calais) pour son aide avec la maintenace du PC sous Linux et les installations des logiciels. Wilfried Montagnier (préparateur TP physique, Calais) nous reste indispensable pour le montage du TP et les fournitures.

2 Réaction chimique oscillante : Préparation chimique

C'est une réaction d'oxydation de l'acide malonique par l'ion bromate. La ferroïne, indicateur redox permet de visualiser les oscillations du potentiel de la réaction.

2.1 Produits

Nom produit	CAS	formule	concentration
acide malonique 99% référence 125 262 500 de ACROS ORGANICS	141-82-2	$COOH-CH_2-COOH$	M=104,06g/mol
acide sulfurique 95–97% référence CL00.2610.1000 de CHEM-LAB	7664-93-9	H_2SO_4	M=98,08g/mol d=1,84g/ml
bromate de potassium 99,5% référence 268 392 500 de ACROS ORGANICS	7758-01-2	$KBrO_3$	M=167,01g/mol
bromure de potassium 99% référence 222 555 000 de ACROS ORGANICS	7758-02-3	KBr	M=119g/mol
ferroïne (1,10-phenanthroline iron (II) sulfate) référence 1.09193.0100 de MERCK			1/40mol/l

*sadovski@univ-littoral.fr

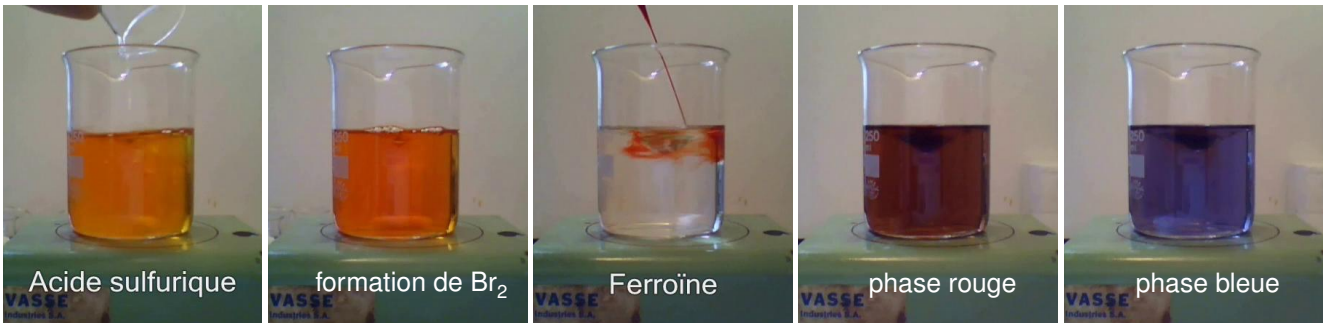


FIG. 1 – Etapes de la préparation de la réaction

2.2 Mode opératoire

On commence le travail dans la salle TP chimie avec les précautions indiquées si-dessous. Quatre solutions sont à préparer

solution A peser 6,75g de bromate de potassium dans 80ml d'eau déminéralisée ($C_{KBrO_3} = 0,5$mol/l)

solution B peser 15,6g d'acide malonique dans 100ml d'eau déminéralisée ($C_{C_3O_4H_4} = 1,5$mol/l)

solution C peser 1,3g de bromure de potassium dans 70ml d'eau déminéralisée ($C_{KBr} = 0,156$mol/l)

solution D prélever 14ml d'acide sulfurique dans 65ml d'eau déminéralisée ($C_{H_2SO_4} = 3,94$mol/l). Notez que l'acide sulfurique disponible à l'ULCO est concentré à 95%. Dans ce cas, prélever un volume de 17ml d'acide sulfurique dans 62ml d'eau déminéralisée[1].

Attention : cette réaction est exothermique, la préparer dans un bain d'eau froide et de glace (verser l'acide dans l'eau) et laisser refroidir. Par ailleurs, il faut une cuve à ultra-son surtout pour la solution de bromate qui est très dure à dissoudre.

Les quatre solutions sont ajoutées les une après les autres et agitées dans un bécher[2] de 500ml muni d'un barreau aimanté et dans un ordre bien précis (A, B, C, D). Cette étape se passe sous une sorbonne, du fait de l'apparition d'une coloration brune, formation de dibrome suite à l'ajout de l'acide sulfurique. Attendre quelques instants, la solution devient incolore.
Après que la solution devient incolore, on peut transférer le bêcher[2] dans la salle TP physique et installer des appareils de mésure (voir la sec. 3.2 et 3.3). Une fois tout est prêt pour effectuer les mesures, verser 2ml de ferroïne et observer.

2.3 Résultat

L'addition de la ferroïne, indicateur coloré, entraine une coloration de la solution en rouge. Après environ 1 minute, la couleur de la solution devient bleue et disparaît en quelques secondes pour de nouveau réapparaître en rouge. Et ainsi de suite, la réaction oscille entre une coloration de la solution en rouge et bleue. Ce phénomène s'observe pendant environ 30 minutes à 1 heure (avec la période d'oscillation beaucoup moins long, 1 à 15 sec).

2.4 La description chimique de cette réaction

Histoire Boris Belousov discovered in the 1950's that an acidic aqueous mixture of malonic acid, potassium bromate, and cerium sulfate in a batch reactor did not react directly to equilibrium. Instead, the composition and color of the solution oscillated for a significant period. His reports of this work were *rejected* by peer-reviewed journals whose reviewers believed that an oscillating chemical reaction *violates thermodynamic equilibrium* and therefore—cannot occur ! Belousov's work did appear in a conference proceeding. Anatolií Zhabotinsky came across this paper in the 1960's, repeated and expanded the work, and was able to publish his research. Par la suite, un très grand nombre des articles scientifiques ont été consacrés à ces systèmes, voir sec. 4.4.

Première étape

$$Br^- + BrO_3^- + 2\,H^+ \rightleftharpoons HBrO_2 + HOBr \tag{2.1a}$$

$$Br^- + HBrO_2 + H^+ \rightleftharpoons 2\,HOBr \tag{2.1b}$$

$$Br^- + HOBr + H^+ \rightleftharpoons Br_2 + H_2O \tag{2.1c}$$

$$Br_2 + HOOCCH_2COOH \rightleftharpoons HOOCCHBrCOOH + Br^- + H^+ \tag{2.1d}$$

L'acide bromique $HBrO_2$ réagit avec une grande quantité de Br^- (la concentration en Br^- diminue). Ceci entraine la bromation de l'acide malonique (dernière étape).

[1] Correction par Madison Huddlestone, L3pc 2018–2019

[2] Afin de ne pas distorter le faisceau optique, il est préférable d'utiliser un flacon où un bêcher en verre de 500ml rectangulaire (donc avec des murs plats parallèles).

Deuxième étape

$$\mathrm{BrO_3^-} + \mathrm{HBrO_2} + \mathrm{H^+} \rightleftharpoons 2\,\mathrm{BrO_2} + \mathrm{H_2O} \tag{2.2a}$$
$$\mathrm{BrO_2} + \mathrm{Fe^{2+}} + \mathrm{H^+} \rightleftharpoons \mathrm{Fe^{3+}} + \mathrm{HBrO_2} \tag{2.2b}$$
$$2\,\mathrm{HBrO_2} \rightleftharpoons \mathrm{HOBr} + \mathrm{BrO_3^-} + \mathrm{H^+} \tag{2.2c}$$

Du fait de la baisse de concentration en ion bromure $\mathrm{Br^-}$, l'acide bromique ne peut plus réagir comme auparavant. Il forme donc, en se combinant avec le bromate $\mathrm{BrO_3^-}$, le radical $\mathrm{BrO_2}$ qui permet l'oxydation des ions $\mathrm{Fe^{2+}}$ en $\mathrm{Fe^{3+}}$

Troisième étape

$$2\,\mathrm{Fe^{3+}} + 2\,\mathrm{HOOCCHBrCOOH} + \mathrm{HOOCCH_2COOH} + 3\,\mathrm{H_2O} \rightleftharpoons 2\,\mathrm{Br^-} + 2\,\mathrm{Fe^{2+}} + 3\,\mathrm{HOOCCHOHCOOH} + 4\,\mathrm{H^+} \tag{2.3}$$

Quand la concentration en $\mathrm{Fe^{3+}}$ commence à devenir importante, $\mathrm{Fe^{3+}}$ réduit $\mathrm{Fe^{2+}}$ avec production d'ions bromures $\mathrm{Br^-}$. L'augmentation de leur concentration stoppe alors la progression de la deuxième phase en favorisant la première. Ainsi les ions bromure $\mathrm{Br^-}$ permettent alors d'initier un nouveau cycle.

2.5 Attention

- Les bromates possédant un fort pouvoir oxydant ils doivent être manipulés avec précaution.
- L'acide sulfurique sera neutralisé au moyen de bicarbonate de sodium.
- L'ensemble du milieu réactionnel pourra être traité avec du bicarbonate puis jeter avec beaucoup d'eau.

3 Mésures et enregistrement

Un fois en salle TP physique, le bécher, toujours muni d'un barreau aimanté, est installé sur une table optique. Il y a deux options pour l'agitation : soit la solution est agitée en permanence, soit on l'arrête dès le démarrage de la réaction (après le premier passage rouge–bleu–rouge).

Les changements de couleur de la solution dans le bécher sont mesurés et enregistrés en temps réel t dans une intervalle $[0 \ldots t_{\max}]$ avec $t_{\max}$ de 10–15 min voir plus. Les mesures sont effectuées par deux méthodes : photométrique (sec. 3.2) et vidéométrique (sec. 3.3). Les données son entrées dans un ordinateur PC.

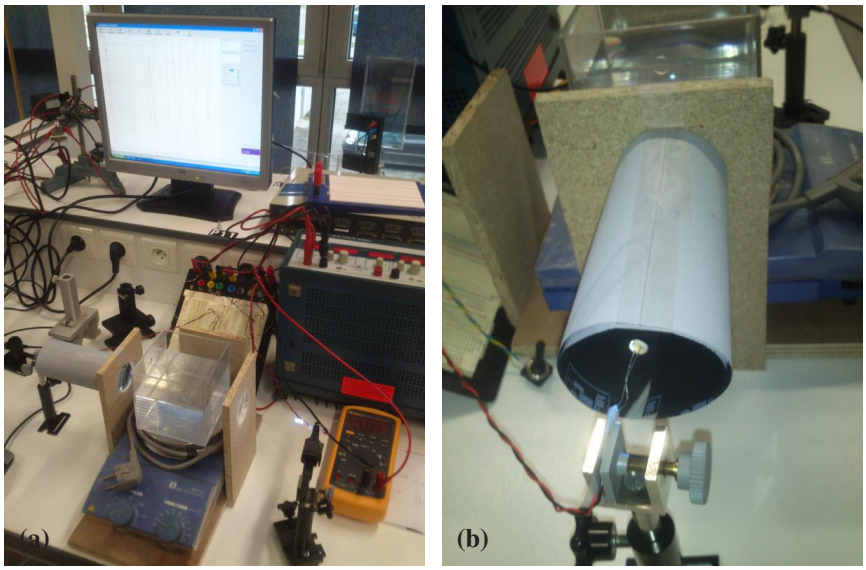


FIG. 2 – Vue générale du montage (a) ; la photorésistance (b) dans le plan focal arrière (AS et DS, 29 nov 2013)

3.1 Matériel et montage

1. deux petites plaquettes, la photorésistance étant soudée sur l'une, l'autre porte le DEL+résistance R_1. Les plaquettess se montent bien dans les pieds optiques et elles sont munis des bornes pour y brancher des cables «banane».
2. une résistance variable de 10kΩ à double molette dans un boitier plexi avec des bornes, voir le potentiomètre P_1, fig. 3(a).

3. la source de la tension DC et/ou du courant continue stabilisée, les branchements se font selon le schéma dans la fig. 3(a).
4. à l'entrée et à la sortie du bêcher[2] (droite et gauche) nous avons placé les lentilles d'une distance focale de 10cm (salle TP optique, diam $\approx$ 40mm). La DEL est mise dans le plan focal avant du système, la photorésistance est placée dans le plan focal arrière. Cela fait collimer un faisceau parallèle de diamètre $\approx$ 40mm qui traverse le bêcher. Ce faisceau augmente la sensibilité et diminue les bruits (en moyennant sur la partie importante du volume de bêcher). Voir la fig. 2(a).
5. Pour sélectionner que la lumière bleue (spectrophotométrie), un filtre peut être placé après la DEL ; pour tester l'enregistrement on utilise un petit rouleau interrupteur de lumière dit «chopper».
6. Les lentilles sont insérées dans les murs lateraux d'un «boîtier» (fig. 2(a)). Ses murs sont noirs, le mur arrière, qui sert de l'écran pour la caméra vidéo, est blanc ; le mur avant est enlevé pour que on puisse filmer la réaction avec la caméra.
7. La photoresistance est insérée dans un tube noir de diam $\approx$ 50mm et de la longueur $\approx$ 150mm collé autour de la lentille de sortie, voir fig. 2(b). Cela protège la photorésistance de la lumière parasite et baisse la ligne de base d'enregistrement photométrique.
8. Le principal est d'exclure la lumière venant de lampes au plafond etc car cela donne 50Hz. On utilise la lumière de jour. Dans l'idéal, le boîtier peut être couvert pour éliminer toutes lumières extérieures et le mur arrière du boitier peut être transformé à un écran en verre diffuseur avec qques DEL's placées derrière face à la caméra et alimentées par la même source du courant continue.
9. dans le boîtier on place le agitateur/mélangeur magnétique (dont on n'utilise que la commande du barreau aimanté) sur lequel on installe le bêcher[2] avec la réaction chimique.
10. un module ORPHY GTI connecté à un PC sous WINDOWS avec les logiciels GTI et RÉGRESSI ; cable GTI style RS-232 (fig. 3(b)), voir sec. 3.2.1.
11. un multimètre en V DC est branché au bornes de la photorésistance pour contrôler la sortie parallèlement au GTI.
12. Une caméra vidéo connectée à un 2me PC sous LINUX avec les logiciels `mplayer` et scripte `mkhisr.sh` ; la caméra est pointée dans le plan horizontal perpendiculairement au faiseau ; à l'arrière, derrière le bêcher, on placera un écran blanc. Un autre écran (noir) est placé derrière la caméra pour éviter tout sortes des reflets parasites provenant de l'extérieur, notamment due aux passages des expérimentateurs pendant les observations.
13. [jan. 2019] Comme une alternative au module ORPHY GTI et son PC dans 10, on utilise un micro-contrôleur ARDUINO UNO chargé avec le programme `adcsampler.ino` (voir les TP ARDUINO, sec. 2.1.6) et connecté au PC sous LINUX par son cable usb, voir sec. 3.2.2. Cela nous permet d'effectuer tout enregistrement et traitement sur le même ordi. L'entrée analogique A0 du micro-contrôleur et sa masse GND se branchent à la place du cable ORPHY GTI, voir fig. 3(a).
14. Une fois tout est prêt et apres les testes du système d'enregistrement photomètrique et vidéo, on ramène un bêcher carré du labo de chimie muni d'un barreau aimanté. Grace à sa forme[2], les faiseaux de la lumière y resteront parallèles. Voir fig. 2(a).

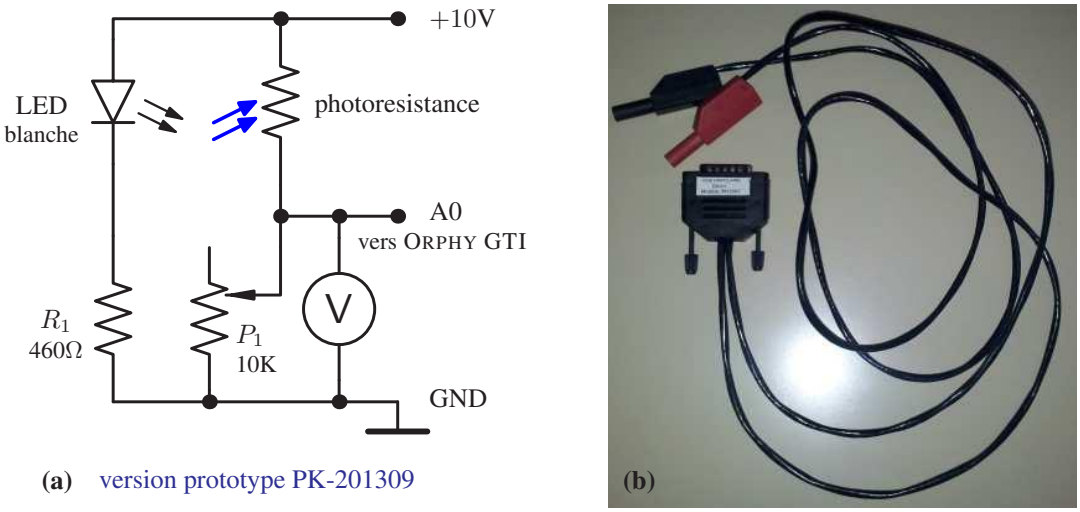


FIG. 3 – Schema électrique du capteur optique (a) avec son cable de branchement ORPHY GTI (b)

3.2 Enregistrement photométrique

On utilise un capteur optique afin de mesurer les variations de couleurs de la solution dans le bécher pendant la réaction oscillante. Le capteur est composé d'une photorésistance et d'une diode électroluminescente (DEL) blanche pour former une barrière lumineuse entre laquelle on va placer notre bécher, voir la figure 3(a) et 4. La résistance ohmique de la photorésistance varie en fonction de

l'éclairement reçu. Un potentiomètre P_1 de 10K placé en série avec cette dernière permet de créer un diviseur de tension, voir la fig. 3(a). Ce montage permet, en mesurant aux bornes du potentiomètre, d'obtenir une tension proportionnelle à l'intensité lumineuse reçu par la photorésistance. Le potentiomètre permet d'étalonner le capteur pour obtenir une tension maximale de $+5$V avant le début de la réaction (en phase rouge). Pour contrôler la tension de sortie P_1 on peut brancher le multimètre.

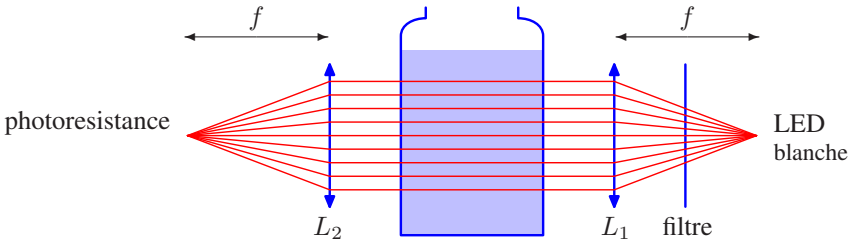


FIG. 4 – Schema optique : la source, le bêcher, le capteur, et le collimateur formé par les deux lentilles de distance focale $f = 10$ cm.

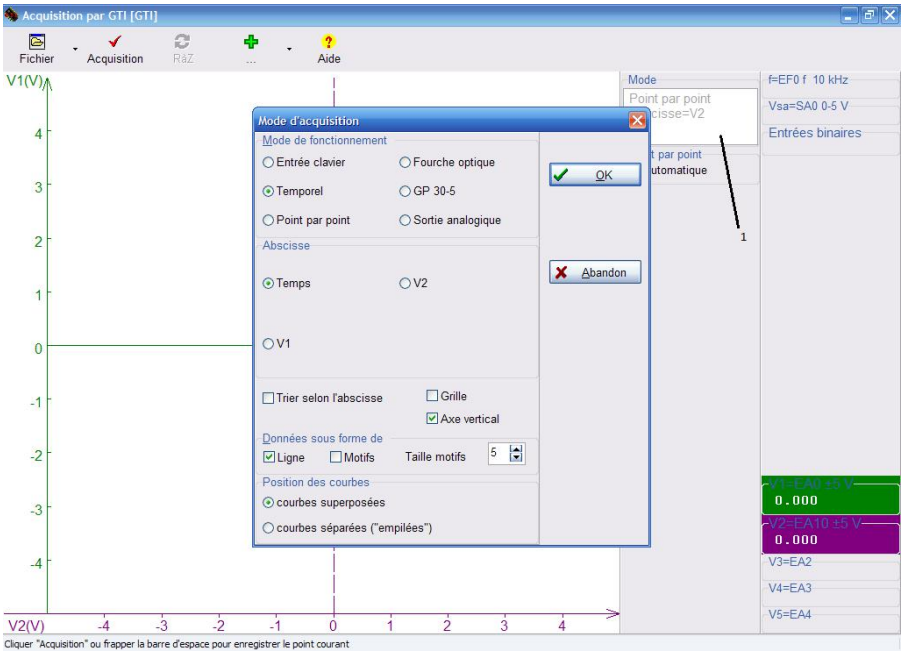
La DEL blanche sert de rétro éclairage. De cette façon, la lumière émise traversant la solution est plus ou moins atténuée en fonction de l'état dans laquelle la réaction se trouve : phase rouge ou phase bleue. Un filtre bleu placé devant la photorésistance ou apres le DEL (fig. 4) permet d'améliorer le contraste lors des changements de phase de couleurs de la réaction et donc d'augmenter l'amplitude du signal lors de l'enregistrement. Notons que d'après nos essais, grâce au rétro-éclairage, le capteur n'est pas trop sensible à la pollution lumineuse environnante car il est saturé par la DEL.

3.2.1 Enregistrement sous WINDOWS avec ORPHY GTI

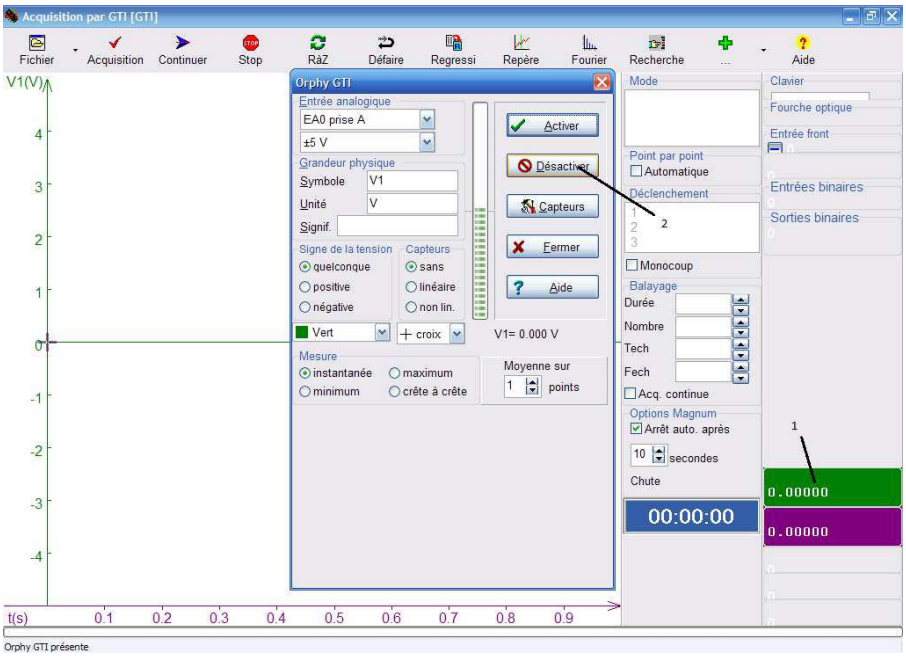
Raccordement du capteur au module ORPHY GTI La sortie du capteur est reliée à un module d'acquisition ORPHY GTI qui convertit le signal analogique provenant du capteur en un signal numérique compréhensible pour le logiciel d'acquisition. La sortie du capteur est reliée au module d'acquisition par un cordon «voie unipolaire», voir figure 3(b), composé d'une prise SUBD15 à relier à l'une des entrée G, H, I ou J du module, et de deux fiches bananes pour brancher le capteur.

Paramétrage du logiciel et étalonnage du capteur Démarrez l'ordinateur, allumez le module ORPHY GTI et ouvrez le logiciel d'acquisition GTI. Pour configurer les paramètres d'acquisition, cliquez dans la fenêtre mode (1) et sélectionnez les valeurs suivantes :

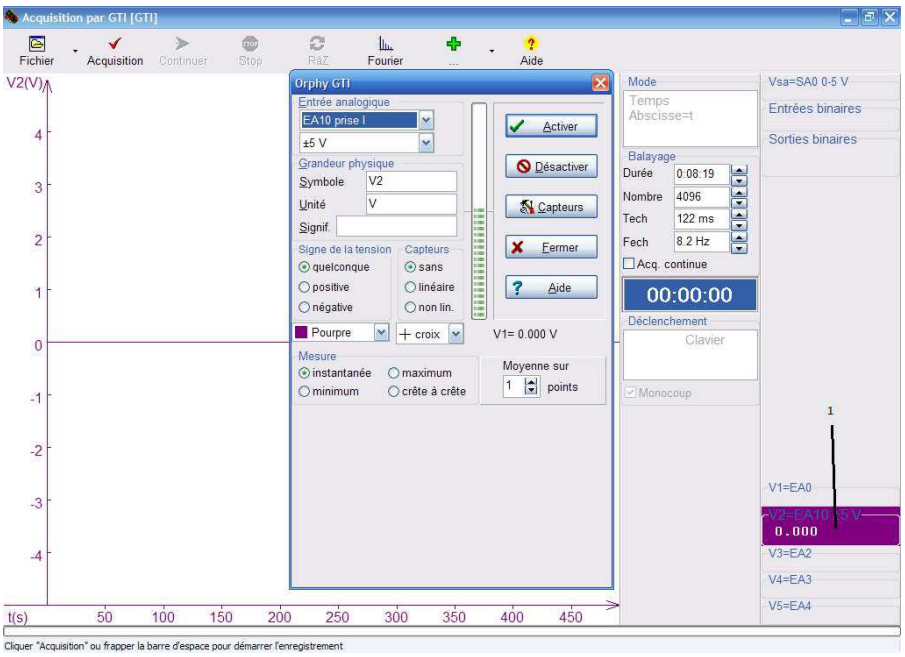
- Mode de fonctionnement : Temporel
- Abscisse : Temps



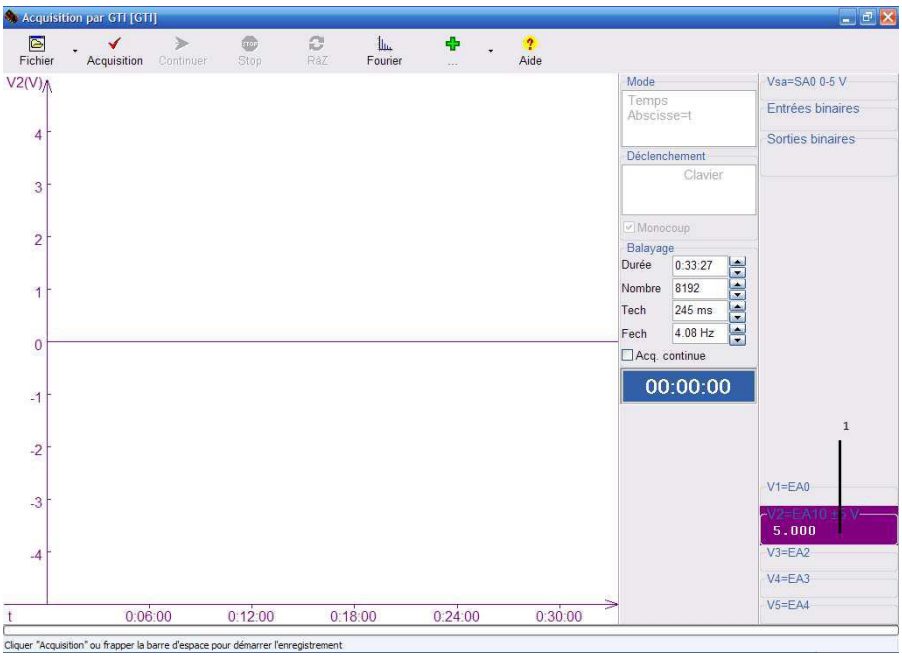
Pour permettre d'enregistrer un grand nombre de points, sur le menu déroulant, sélectionnez «options». Dans les nombres maximales de points, sélectionnez 8192. Par la suite, cliquez sur la fenêtre d'acquisition de la tension V1 (fenêtre verte en bas à droite (1)) et désactivez la (2) car elle ne sera pas utilisée :



Il faut configurer le logiciel pour lui indiquer sur quelle entrée nous avons branché le capteur et la plage de tension à mesurer. Pour cela, cliquez sur V2 (1) et dans les menus déroulant «Entrée analogique», sélectionnez la prise sur laquelle est relié le capteur (G, H, I ou J), et sélectionnez +5V (positive) comme tension d'acquisition.



Etalonner ensuite la tension de sortie du capteur. Pour cela il faut utiliser le potentiomètre afin d'amener la tension V2 à 5V (1) :



Pour terminer, vous devez paramétrer la durée d'acquisition et la fréquence à laquelle le logiciel va mesurer la tension à la sortie de capteur.

Teste fonctionnement du capteur et son interface Une fois le module ORPHY GTI et le logiciel d'acquisition GTI sont en route, et *avant* de ramener le bêcher avec la réaction chimique, on procède à un teste en plaçant un rouleau interrupteur dit «chopper» devant le DEL pour découper la lumière périodiquement (la fréquence de modulation est donnée par la vitesse de rotation). On doit être capapble d'observer et d'enregistrer la fonction «carrée» à la sortie du capteur. On contrôle aussi la tension au bornes du potentiomètre P_1, voir la fig. 3(a), pour ne pas dépasser $+5$V DC.

3.2.2 Enregistrement sous LINUX avec ARDUINO UNO

Branchez le micro-contrôleur ARDUINO UNO sur le port USB du PC sous LINUX. En utilisant deux fiches banane munis des fils fins, connectez l'entrée analogique A0 et la masse selon le schéma dans la fig. 3(a) (à la place de ORPHY GTI). Sous LINUX, après le *login*, on lance un terminal (en cliquant sur l'icone correspondant) et on travaille dans un *shell* exécutant les commandes interactives[3]. A l'occasion il s'agit de `bash` lancé automatiquement à l'ouverture du fenêtre de terminal. Les commandes sont entrées sur la ligne en bas de ce fenêtre après le «prompt» marqué ci-dessous par le caractère >. La commande suivant nous affiche ce manuel sur l'écran

```
> evince reaction_oscillante.pdf &
```

Nous pouvons vérifier la présence de fichiers et programmes nécessaires,

```
> ls -lhtr mkhisr.sh adcsampler.ino reaction_oscillante.pdf
> which gnuplot maxima mplayer arduino
```

et afficher la racine du répertoire de l'utilisateur où on se retrouve après login :

```
> pwd
```

Les commandes

```
> export PATH=$PATH:.
> export srcdir=`pwd`
```

nous faciliterons l'accès au scripte `mkhisr.sh` installé dans la racine `` `pwd` ``. Pour s'assurer que vos fichiers ne se mélangent pas avec ceux d'autres utilisateurs, choisissez le nom de la sous-racine selon votre nom et la date, ou bien selon tout autre format unique, et mémorisez le dans la variable `fnbase`

```
> export fnbase="NomTPYYMMDD"
```

Ces commandes sont executées une fois au début de la session. Au présent, nous (re)chargeons, si nécessaire, le micro-contrôleur Arduino UNO avec le programme `adcsampler.ino` (échantillonnage ADC, voir appendice B) en faisant appel à

```
> mkhisr.sh loadADC
```

et nous vérifions son fonctionnement avec la commande

```
> ../mkhisr.sh testADC
```

En modifiant l'exposition de la photorésistance, on observe les changements respectifs des valeurs affichées s'approchant 1023 (soit 5V) pour l'exposition maximale. Alternativement, et surtout si on veut se familiariser d'avantage avec le fonctionnement du micro-contrôleur, suivre appendice B.

En sachant que les périodes de phénomènes des oscillations qui nous intéressent sont d'ordre de $T_{\rm osc} \approx 10$ sec, que le nombre d'échantillons par période doit être assez important, entre 10 et 100, et que le nombre total d'échantillons est souhaitable (pour la transformation Fourier rapide par la suite) d'être choisi proche à une puissance de 2, telle que 512, 1024, 4096, tout en restant dans le raisonnable, définissez la période d'échantillonnage en μsec et le nombre des échantillons, puis lancez la commande de l'enregistrement

```
> ADCTIME=100000 ADCNPTS=1024 ../mkhisr.sh sampleADC
```

Ici on a choisi d'échantillonner à la fréquence de 10Hz, donc avec une centaine des points par $T_{\rm osc}$ pendant une dizaine des oscillations. On attendra, alors, 100 sec jusqu'à la fin de l'enregistrement. Les données sont sauvegardées dans un fichier texte dans la racine d'utilisateur (`` `pwd` ``) sous le nom designé par `fnbase` avec une extention `-ADC.dat`, soit `NomTPYYMMDD-ADC.dat` selon la choix suggérée ci-dessus. Le format de ce fichier est simple : une colonne des échantillons de valeurs 0..1023 et quelques commentaires marquées par #.

3.3 Enregistrement vidéométrique sous LINUX

En même temps, le bêcher est filmé par une caméra vidéo, dont l'axe de la vision est perpendiculair au faisceau décrit dans la sec. 3.1 et 3.2. La caméra est montée devant le bêcher. Derrière le bêcher, dans le champ de vision de la caméra, on place un écran blanc mat. La caméra est connectée à un 2$^{\rm e}$ PC sous LINUX par le port USB et la séquence vidéo est y enregistrée sur le disque dur avec une vitesse de 12 ou 24 cadres par seconde[4] (à prevoir environs 100M d'espace disque). Les aspects généraux du travail sur le PC sous LINUX sont déjà décrits dans la sec. 3.2.2. Pour tester le fonctionnement de la caméra vidéo et visualiser son alignement, lancez

```
> mplayer tv://
```

NB : on termine `mplayer` (où toute autre commande lancée par `bash`) par un `Ctrl-C` dans le fenêtre de terminal. On peut terminer `mplayer` en tapant `q` (pour *quit*) dans l'écran vidéo. Au début de la session, vous pouvez aussi mémoriser le nom du fichier vidéo (ici `NomTPYYMMDD.avi`) avec l'extention `.avi`.

[3] le shell sert ainsi d'un interprète des commandes, dit CLI=command line interpreter ; sous WINDOWS ceci est connu comme «mode DOS».
[4] on utilise souvent l'abréviation anglaise fps = frames per second

```
> export movief=$fnbase.avi
```

Au présent, pour tester la caméra vous pouvez passer par

```
> mkhisr.sh video
```

Pour commencer l'enregistrement[5] lancez

```
> RECFPS=8 mkhisr.sh record
```

Au bout de quelques minutes, terminez l'enregistrement par un `Ctrl-C`. Vous pouvez visualiser et vérifier votre enregistrement avec

```
> mplayer $movief
```

4 Traitement des données

4.1 Données photométriques

En utilisant ORPHY GTI (sec. 3.2.1), les données photométriques arrivent dans le PC sous WINDOWS déjà sous forme exploitable, comme un tableau des données $[t, I_R(t), I_B(t)]$. On peux les tracer en utilisant `Régressi` et, par la suite, passer directement à l'analyse Fourier dans la section 4.3. A noter, que cet analyse peut être effectuée avec `Régressi`.

Dans le cas des mésures effectuées par l'ADC A0 de ARDUINO gérées par le PC LINUX (sec. 3.2.2), il est préférable de garder vos fichiers dérivés dans un sous-répertoire nommé `fnbase`, voir la sec. 3.2.2. Ce sous-répertoire peut être créé par la commande (lancée depuis la racine d'utilisateur !)

```
> mkdir -p $fnbase; cd $fnbase
```

Au present, étant dans le sous-répertoire `fnbase`, lancez

```
> nmin=0 nfft=10 ../mkhisr.sh ADCplot; evince ADCplot.pdf
```

pour obtenir et visualiser la représentation graphique de vos données. Ici, les paramètres optionnels `nmin` et `nfft` donnent la possibilité de définir (réduire) l'intervalle de visualisation des données, à commencer et à terminer avec l'échantillon `nmin` $+1$ et `nmin` $+2^{\texttt{nfft}}$, respectivement (de 1 à 1024 dans l'exemple ci-dessus). On utilisera les mêmes options et le même intervalle pour l'analyse Fourier dans la sec. 4.3. Sauvegardez le fichier `ADCplot.pdf` et les données ADC (extention `-ADC.dat`) sur une clé usb pour les exploiter ailleurs. Vous pouvez revenir dans la racine d'utilisateur avec

```
> cd ..
```

4.2 Séquence vidéo

Les données $[n, I_R(n), I_G(n), I_B(n)]$, où $n = 1 \ldots n_{\max}$ est le numéro du cadre lié au temps réel par

$$t = n/f_{\text{fps}},$$

doivent être extraites de la séquence vidéo enregistrée (voir la sec. 3.3). Pour cela on utilise un ordinateur assez puissant avec un espace disque additionnel de 400M (soit 500M en tout si le vidéo est enregistré sur le méme disque dur).

4.2.1 Procedure sous systèmes LINUX, MACOS, ou d'autres UNIX

Logiciels En plus des utils standards, tels comme `bash`, `sed`, et `awk`, on doit avoir le décodeur vidéo `mplayer`, pour visualiser le filme et pour extraire ses cadres ; la bibliothèque des programmes `netpbm` pour manipuler les images extraites ; `gnuplot` pour tracer les courbes, un bon vieux `xv`, ou `xzgv`, ou tout autre programme pour vérifier le cadrage de la zone de prélèvement des couleurs. Notez le `ffmpeg` comme une alternative à `mplayer`. Tout ces logiciels sont libres est assez courants ; on les trouve préinstallés dans la plupart des distributions. Enfin, pour effectuer la transformation Fourier, on puisse passer par MAXIMA, le logiciel libre très puissant du calcul symbolique et numérique qui possède la bibliothèque `fft`, et on peut utiliser l'interface de MAXIMA pour tracer avec `gnuplot`.

Mode de travail Sauf pour quelques visualizations (la zone de prélèvement, le vidéo même si on veut, les graphs avec les courbes extraites), le travail principal est fait dans un fenêtre du terminal avec un shell préféré, ici soupposé `bash`. Tous les opérations sont gérées par le scripte `mkhisr.sh`, voir l'Appendice A.

Les cadres vidéo sont nombreux et il est pratique de les garder dans le répertoire nommé `fnbase`, voir la sec. 3.2.2. Ce répertoire doit déja exister après le traitement des données photométriques, mais on peut s'assurer de son existence et le créer, si nécessaire, par la commande (lancée depuis la racine d'utilisateur !)

```
> mkdir -p $fnbase; cd $fnbase
```

[5] Avant de commencer l'enregistrement terminez `mplayer` en mode test

Extraction des cadres Pour extraire les cadres (frames) individuels dans ce nouveau répertoire, lancez

```
> RECFPS=8 ../mkhisr.sh frames
```

Cela peut prendre quelques minutes. Quand l'extraction est terminé, vous pouvez vérifier que les fichiers `.jpg` sont crées

```
> ls
```

et vous pouvez visualiser les cadres extraits en lancant le logiciel `xv`

```
> xv 000*.jpg &
```

Dans `xv` utilisez `Space` et `BackSpace` pour naviger vos fichiers (cadres), la touche [`i`] pour ouvrir le fenêtre d'information (taille de cadre et du carré «zoom»), et la touche [`q`] pour quitter. Le carré de «zoom» est activé en appuyant sur la touche gauche de la souris.

Vérification de la sélection de l'échantillon Le carré dont les pixels sont utilisés pour extraire l'information sur les couleurs peut être spécifié par ses coordonnées `(X0,Y0)` et sa taille `(XX,YY)` en pixels. Ces quatre nombres entiers peuvent être déterminés en utilisant `xv`. Dans une image choisie, par exemple le fichier numéro 200, placez le carré de «zoom» pour sélectionner la partie de bêcher où le milieux réactionnel se présente assez homogène (donc sans abords etc). Notez les coordonnées affichées dans le fenêtre des informations, par exemple `X0=303 Y0=103 XX=148 YY=190` et lancez la commande[6]

```
> RECFPS=8 crop_X0=303 crop_Y0=103 crop_XX=148 crop_YY=190 sample_frame=200 ../mkhisr.sh sample
```

Cela va découper (affleurer) le carré choisi et extraire l'échantillon dans le fichier `sample.jpg` qu'on vérifie avec

```
> xv sample.jpg
```

Données RGB $[n, I_{\mathsf{R}}(n), I_{\mathsf{G}}(n), I_{\mathsf{B}}(n)]$ Le programme `mkhisr.sh`, voir l'Appendice A, découpe le carré de l'échantillon de chaque cadre $n = 1 \ldots n_{\max}$ (chaque fichier $\langle n \rangle$`.jpg` trouvé dans le répertoire). Pour chaque point k (dit «pixel») de l'échantillon ce programme décode $(I_{\mathsf{R},k}, I_{\mathsf{G},k}, I_{\mathsf{B},k})$, un triple des nombres entiers positifs avec les valeurs dans $0 \ldots 255$ (un byte), et prends la moyenne pour chaque canal R=rouge, G=vert, et B=bleu sur tout les pixels de l'échantillon

$$I_{\mathsf{R}}(n) = \sum_k I_{\mathsf{R},k}/k_{\max}, \quad I_{\mathsf{G}}(n) = \sum_k I_{\mathsf{G},k}/k_{\max}, \quad I_{\mathsf{B}}(n) = \sum_k I_{\mathsf{B},k}/k_{\max}.$$

On obtient ainsi $[n, I_{\mathsf{R}}(n), I_{\mathsf{G}}(n), I_{\mathsf{B}}(n)]$. Cette procédure est lancée par la commande[6]

```
> RECFPS=8 crop_X0=303 crop_Y0=103 crop_XX=148 crop_YY=190 sample_frame=200 ../mkhisr.sh histo > RGB.dat
```

Cela peut prendre du temps ... on patiente. Les données dans le fichier `RGB.dat` peuvent être visualisées en utilisant[6]

```
> RECFPS=8 crop_X0=303 crop_Y0=103 crop_XX=148 crop_YY=190 sample_frame=200 ../mkhisr.sh plot
```

pour produire le fichier `RGBplot.pdf` qu'on affiche avec

```
> evince RGBplot.pdf
```

voir la fig. 5(a). Le fichier `RGB.dat`, un simple fichier texte à quatre colonnes, ainsi que `RGBplot.pdf` et la séquence vidéo `NomTPYYMMDD.avi` peuvent être saufgardés sur une clé usb pour être exploité ailleurs. Tout autres fichiers peuvent être effacés.

4.2.2 Procedure sous système WINDOWS

Si on y tient vraiment ... Le logiciel `mplayer` et la bibliothèque `netpbm` sont disponibles également sous WINDOWS. Cependant il faut trouver un langage qui permet organiser un boucle. Le MS-DOS de WINDOWS possède un langage de script rudimentaire dit «`batch`» semblant à celui de `sh`. Si non, VISUAL BASIC ou d'autres bidouilles. La représentation graphique peut elle aussi effectuée avec `gnuplot`, mais à ce point, on peut basculer tout simplement vers `Régressi`.

4.3 Spectre Fourier, analyse et observations

Le reste ... Dans la sec. 4, nous avons décrit les étapes incontournables. Le reste peut être effectué en basculant vers `Régressi` (libre) sous WINDOWS, ou en utilisant d'autres logiciels (commerciaux), par exemple MAPLE, MATHEMATICA, MATLAB, capables à la fois de tracer les graphes et d'effectuer la transformation Fourier. Ceux qui préfèrent de continuer avec les logiciels libres sous LINUX ou MACOS, peuvent installer le scripte `mkhisr.sh` (sous `bash`) et utiliser ces différents commandes, voir l'Appendice A.

[6] Notez que les commandes exécutées précédemment par l'interprète `bash` peuvent être rappelées avec la touche «flèche en haut» et modifiées en utilisant les flèches gauche-droite et l'efasseur. Ainsi on évite de retaper entièrement les longues commandes.

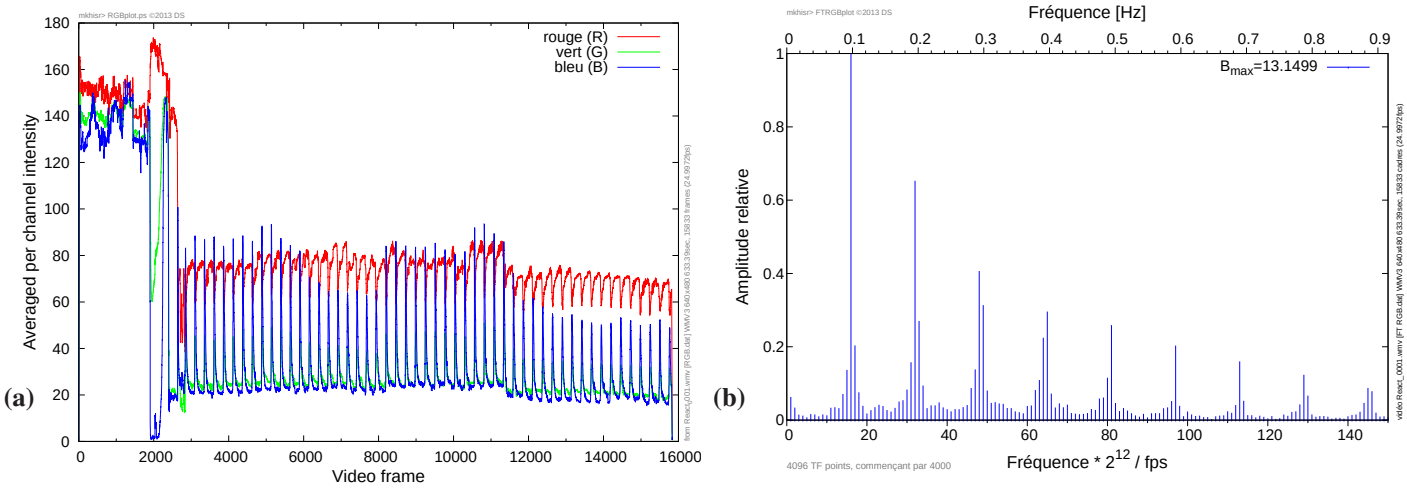


FIG. 5 – (a) Données sur les intensités RGB extraites de la séquence vidéo ; (b) Transformation Fourier de l'intensité $I(t)$ de la composante bleue B dans (a). L'échantillon de TF est de 4096 cadres commençant par le cadre 10000.

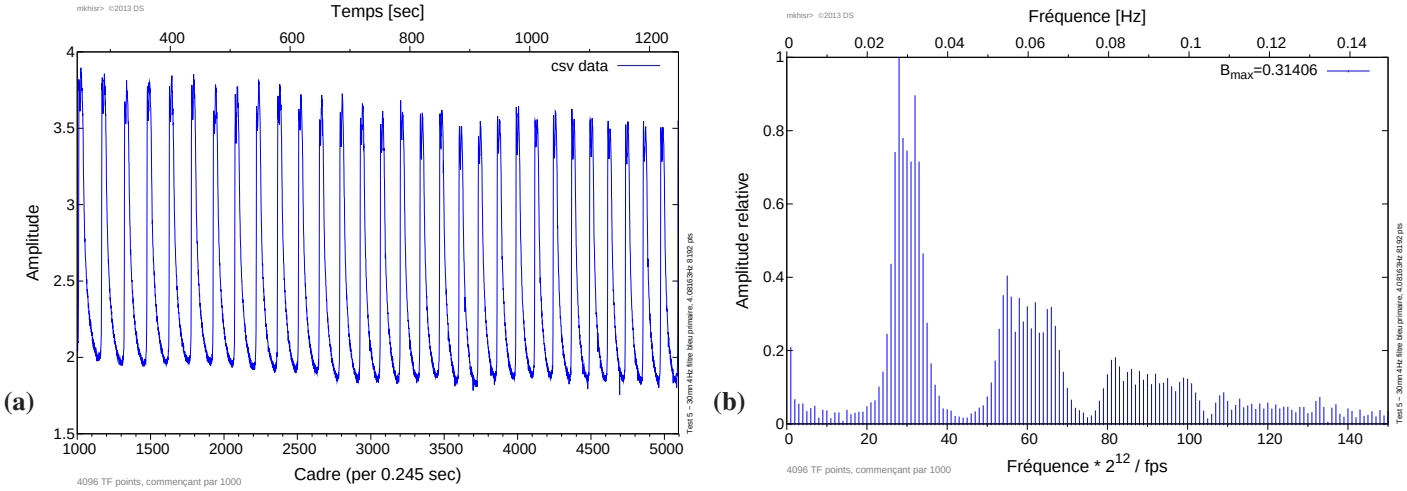


FIG. 6 – (a) L'intensité $I_B(t)$ enregistrée par le capteur optique muni du filtre bleu ; (b) la transformée Fourier de $I_B(t)$; l'échantillon (a) est de 4096 cadres commençant par le cadre 10000.

Transformation Fourier Pour obtenir et visualiser la transformée de Fourier (TF) de vos données vidéo RGB on utilise[6]

```
> nmin=1 nfft=10 plotfmax=128 ../mkhisr.sh FTplot
```

Les options ici impliquent qu'on analyse les $2^{10} = 1024$ premiers échantillons dans `RGB.dat` pour la TF, et que le graphique s'arrête au point 128 (fréquence) du spectre. Le graphique est sauvegardé dans le fichier `FTRGBplot.pdf`, et, comme auparavant, on l'affiche avec `evince`, voir la fig. 5(b). Pour vos données photométriques obtenues avec ARDUINO UNO, voir la fig. 6(a), on utilise[6]

```
> nmin=1 nfft=10 plotfmax=128 ../mkhisr.sh FTADCplot
```

afin de créer le fichier `FTADCplot.pdf`, voir la fig. 6(b).

Questions Quelle est la période des oscillations $T_{\rm osc}$? Leur forme ? Ce phénomène, est il linéaire ? atténué ? Si oui, trouvez la constante d'atténuation γ. Pourquoi observe-t-on les harmoniques ? Que pouvez vous constater concernant la dépendance de $T_{\rm osc}$ et γ de la température et la vitesse de mélangeur. Proposez (recherchez) les explications.

pour DS : La période n'est pas la même dans les figures 5(b) et 6. Quelle est la raison ? Experience avec/sans agitateur ? Temperature ?

4.4 Modélisation

On considère les équations différentielles de la cinétique chimique décrivant les réactions (2.1), (2.2), et (2.3), aussi bien que le transport vers/de la zone de la phase catalytique (faites un recherche bibliographique).

Ci-dessous nous citons le mécanisme FKN (Field–Körös–Noyes) pour la réaction B-Z catalysée par Ce^{4+}. Se mécanisme est réduit à 5 réctions dans le modèle OREGONATOR. Il faut obtenir le système des équations pour les reactions dans la sec. 2.4 avec la participation de Fe^{3+} (ferroine)

Apres simplifications, il s'agit d'un système à trois équations différentielles ordinaires nonlinéaires couplées d'ordre 1, donc à trois degrés de liberté (3-species model).

$$\begin{aligned}
\dot{x}(t) &= k_1 A\, y(t) - k_2 x(t) y(t) + k_5 A\, x(t) - 2k_6 x(t)^2, \\
\dot{y}(t) &= -k_1 A\, y(t) - k_2 x(t) y(t) + k_7 f\, z(t), \\
\dot{z}(t) &= k_5 A\, x(t) - k_7 z(t),
\end{aligned}$$

où $x(t)$, $y(t)$ et $z(t)$ représentent la concentration en acide bromeux $HBrO_2$, en ions bromure Br^- et en cerium-IV Ce^{4+} respectivement ; $A = 1$ est la concentration initiale en bromate BrO_3^- (invariante lors de la réaction), $f \in 0.5 \ldots 2.4$ est un paramètre stoichiometrique ajustable, et

$$k_1 = 0.05,\ k_2 = 0.8,\ k_5 = 1.58,\ k_6 = 0.2,\ k_7 = 0.4$$

sont des constantes de vitesse de réactions connues. La solution oscillatoire $z(t)$ ne peut pas être obtenue que numériquement et elle est très sensible aux paramètres du système (vitesses des réactions, température, etc) ainsi qu'aux conditions de départ. Il est envisageable d'obtenir une telle solution avec un intégrateur RK de MAXIMA et l'étudier.

A Programme de traitement des séquences vidéo sous LINUX

Le script `mkhisr.sh` ci-dessous peut être utilisé sous LINUX, MACOS, ou sous autres systèmes du type UNIX pour extraire les cadres d'une séquences vidéo, faire la transformation Fourier et pour tracer les graphs. Voir la section 4.2.1 pour des explications, et la fig 5 pour les graphiques qu'il produit.

Definitions Définirons quelques variables du `shell`

```
#!/bin/bash
  wrkdir=${wrkdir:-$PWD}            # répertoire de travail
  srcdir=${srcdir:-/mnt/cdrom}      # répertoire où le vidéo se trouve
  movief=${movief:-React_0001.wmv} # le nom du fichier pour le vidéo
  movie=$srcdir/$movief             # le nom entier
```

Par la suite, si on travail dans un shell interactif, faisons

```
  export wrkdir srcdir movief movie
```

Informations sur la séquence vidéo Tout d'abord on fait appel à `mplayer` pour collecter les informations sur le vidéo :

```
  mplayer -identify -nosound -benchmark -frames 0 $movie
```

Cela sorte un nombre des caractéristiques, parmis lesquelles `ID_LENGTH` donnant la durée en secondes nous intéresse. On note aussi `ID_VIDEO_FORMAT`, et les dimensions du cadre `ID_VIDEO_WIDTH`, `ID_VIDEO_HEIGHT`. Insertons les tous comme variables (sans préfix `ID_`)

```
  eval 'mplayer -identify -nosound -benchmark -frames 0 $movie | awk '/ID_[^=]+=/ {print $0}' '
```

Extraction des cadres Il souffit de taper

```
  mplayer -quiet -nosound -benchmark -vo "jpeg:outdir=$wrkdir" $movie
```

pour obtenir tout les cadres sous format `jpeg`. Leur noms seront du type `00000001.jpg` et leur nombre est donné par

```
  ls -1 $wrkdir/0*.jpg | wc -l
```

Dans un script, il nous sera utile definir une fonction correspondant

```
frame_count ()
{
    ls -1 $wrkdir/0*.jpg | wc -l
}
```

Ainsi dans notre test démo `React_0001.wmv`, il y a 15 833 cadres de taille totale 372M. La durée est de 633.39 secondes.

La zone de prélevment Il nous faut definir une zone rectangulaire dans les images où les couleurs RGB seront mesurées.

```
  crop_X0=${crop_X0:-346}
  crop_Y0=${crop_Y0:-188}
  crop_XX=${crop_XX:-24}
  crop_YY=${crop_YY:-85}
```

Ainsi nous définissons les coordonnées par défaut d'une zone de 24×85 pixels commençant à $(346, 188)$. Cette zone était utilisée pour notre test démo. Pour trouver ces valeurs on peut visualiser l'un des cadres avec `xv` et utiliser son fonction «`crop`» pour tailler la zone désirée, puis ouvrir le fenêtre «`info`».

Les couleurs RGB pour un cadre de numéro `$1` nous trouvons le nom de son fichier `.jpg` avec

```
set_ffname ()
{
    echo $1 | awk '{printf("%08i.jpg",$1)}'
}
```

Par la suite, en utilisant un fichier d'échantillon intermédiaire

```
  sample_file="sample.jpg"
```

nous faisons les opérations suivantes

```
frame_RGB ()
{
    local i C0 C1 C2
    jpegtopnm `set_ffname $1` 2>/dev/null | pamcut  $crop_X0 $crop_Y0 $crop_XX $crop_YY | pnmsmooth >
        $sample_file 2>/dev/null
    for i in `seq 0 2`; do
  eval C${i}=`cat $sample_file | pamchannel $i |  pamsumm -mean | sed -e 's/^[^0-9]*//'`
    done
    i=`echo $1 | awk '{printf("%8d",$0)}'`
    echo "$i" $C0 $C1 $C2
}
```

Ici après avoir transformé le `.jpg` en `.pnm`, avoir extrait la zone avec `pamcrop`, et l'avoir lissé avec `pnmsmooth` (optionnel), nous sortons chaque «canal» 0,1,2 correspondant à R,G,B avec `pamchannel` et obtenons l'intensité du canal comme la somme sur tout les pixels de la zone avec `pamsumm`.

Données RGB pour chaque cadre L'opération précedante ce fait en boucle suivant

```
  fmax=`frame_count`;
  echo "#  frame  R          G          B"
  for (( f=1 ; f<=fmax ; f++ )) ; do
      frame_RGB $f
      (( f%100==0 )) && echo "   analyzed $f frames of $fmax" > /dev/stderr
  done
```

Vu le grand nombre des cadres `fmax` ce boucle prendra beacoup de temps (sur un PC de 1–2GHz, les temps d'ordre 15–30min à anticiper). Pour indiquer la progréssion, nous faisons si-dessus un message pour chaque 100 cadres.

Programme complet

```
#!/bin/bash
#-------------------------------------------------------------------------------
# This should work on Linux/MacOS with awk, mplayer and netpbm installed
# requires maxima for Fourier transform, and gnuplot for plots
#
# NB: mplayer has the ability to start at a given time (-ss option) and extract
# just one or several frames ( -frames option), but I have never made it work
#
# To determine the area of color sampling (should be done once) one can use any
# image viewer that allows cropping such as the old xv on linux/unix, or do a
# series of trial crops using netpbm's routines ...
#
# see help for instructions
#
# DS <sadovski at univ-littoral.fr> 20130907 20190124 (Arduino) 20190213
#-------------------------------------------------------------------------------
VERSION="20201123 (Arduino)"
PWD=`pwd`
USER=${USER:-`who -m | cut -f 1 -d" "`}
```

```
HOST=${HOST:-` uname -n`}

fnbase=${fnbase:-TP$USER`date +%y%m%d`}
wrkdir=${wrkdir:-$PWD}           # /mnt/scratch/tmp/test
srcdir=${srcdir:-/mnt/cdrom}     # folder where the movie file is kept
movief=${movief:-React_0001.wmv} # file name with the movie
movie=$srcdir/$movief            # complete path to the movie file
adcdata="$srcdir/$fnbase-ADC.dat" # complete path to data from Arduino ADC
# NB: default file type for recording is avi with mpeg4 ffmpeg codec
ADCTIME=${ADCTIME:-100000}       # ADC sampling period in usec
ADCNPTS=${ADCNPTS:-1024}         # total number of ADC samples
RECFPS=${RECFPS:-4}              # FPS (frames per second) rate for recording
# gnuplot in the enhanced PostScript mode has a special use for the underscore
# NB: \\\\\\\\ results just in a single backslash in the gp script at the end ...
gpmovief=`echo $movief | awk '{gsub(/_/,"\\\\\\\\\\\\\\\\\\_"); print $0}'`; # echo $gpmovief
if [ -r "$movie" ]; then
    # interrogate the actual movie file for information
    eval `mplayer -identify -nosound -benchmark -frames 0 $movie 2>/dev/null | \
       awk -F '=' '/ID_[^=]+=/ {print $1 "=" "\"" $2 "\""}'`
# NB: gawk doesn't work on Debian (and their awk doesn't know gensub) ...
# as an alternative, could use sed which understands \1 \2 regexp replacements
else
    if [ -r README.frames ]; then
        # recover the information on the previous frame extractions
  eval `awk '/ID_[^=]+=/ {gsub(/ /,"_"); print $0}' README.frames`
    else
        # makes available default values of the following variables
  ID_AUDIO_ID=1
  ID_VIDEO_ID=2
  ID_CLIP_INFO_NAME0=title
  ID_CLIP_INFO_VALUE0="Reaction oscillante"
  ID_CLIP_INFO_NAME1=author
  ID_CLIP_INFO_VALUE1="Pierre Kulinski"
  ID_CLIP_INFO_NAME2=copyright
  ID_CLIP_INFO_VALUE2="2013"
  ID_CLIP_INFO_NAME3=comments
  ID_CLIP_INFO_VALUE3="premier essai effectutue par Pierre et Amory"
  ID_CLIP_INFO_N=4
  ID_FILENAME=/mnt/cdrom/React_0001.wmv
  ID_DEMUXER=asf
  ID_VIDEO_FORMAT=WMV3
  ID_VIDEO_BITRATE=1900000
  ID_VIDEO_WIDTH=640
  ID_VIDEO_HEIGHT=480
  ID_VIDEO_FPS=1000.000
  ID_VIDEO_ASPECT=0.0000
  ID_START_TIME=1.58
  ID_LENGTH=633.39
  ID_SEEKABLE=1
  ID_CHAPTERS=0
  ID_VIDEO_CODEC=ffwmv3
  ID_EXIT=EOF
    fi
fi
csvfile=${csvfile:-"Aquisitions react oscii/Test 5 - 30mn 4Hz filtre bleu primaire.csv"}
# TODO: should it correlate with movief ?
plotf="RGBplot.ps"
dataf="RGB.dat"
adcplotf="ADCplot.ps"
nmin=${nmin:-4000}        # start point (frame) for the FT sample interval
nfft=${nfft:-12}          # 2^nfft is the number of sample points for FFT
plotfmax=${plotfmax:-150} # maximum FT point to be plotted

#   React_0001.wmv, 15833 frames, 372M in /mnt/scratch/tmp/test/
#   for this movie we have to crop to a 24x85 rectangle starting at 346,188
crop_X0=${crop_X0:-346}
crop_Y0=${crop_Y0:-188}
crop_XX=${crop_XX:-24}
crop_YY=${crop_YY:-85}
#   (X0,Y0)=(left,top) (X1,Y1)=(right,bottom) (XX,YY)=(width,height)
(( crop_X1 = crop_X0 + crop_XX ))
(( crop_Y1 = crop_Y0 + crop_YY ))

OURPROMPT=`basename $0 .sh`"> "
if [ -z "${ID_LENGTH}" ]; then
    echo $OURPROMPT"cannot determne the duration (ID_LENGTH) of the movie file!!"
```

```
else
    echo $OURPROMPT"movie file $movie ${ID_LENGTH}sec long"
fi
# set jpeg file name correspoding to the sequential index of the frame
# used by mplayer, e.g. 12668 becomes 00012668.jpg
set_ffname ()
{
    echo $1 | awk '{printf("%08i.jpg",$1)}'
}
# number of the frame used for the illustration of the sample rectangle
sample_frame=${sample_frame:-9498}
sample_file=sample.jpg

# total number of frames (indexes starting with 1) in the working directory
frame_count ()
{
    ls -1 $wrkdir/0*.jpg | wc -l
}

# the FPS supplied by mplyer is bogous, should compute our own (about 24-25)
frame_info ()
{
    echo $OURPROMPT"Working directory is $wrkdir"
    echo $OURPROMPT"The disk space used:" `du -sh $wrkdir | awk '{print $1}'`", hope you can live with this ..."
    echo $OURPROMPT"There is total of" `frame_count` "frames in" `basename $movie`
    echo $OURPROMPT"Video format $ID_VIDEO_FORMAT ${ID_VIDEO_WIDTH}x${ID_VIDEO_HEIGHT} ${ID_LENGTH}sec"
}

# for a given frame, return mean R,G,B channel values in its sample area
# hmm? think this is mean intensity value for each channel ...
# alternatively can use   cat $sample_file | ppmhist | do smth using awk ...
# might also take a look at pnmgamma?
frame_RGB ()
{
    local i C0 C1 C2
    jpegtopnm `set_ffname $1` 2>/dev/null | pamcut  $crop_X0 $crop_Y0 $crop_XX $crop_YY | pnmsmooth >
        $sample_file 2>/dev/null
    for i in `seq 0 2`; do
  eval C${i}=`cat $sample_file | pamchannel $i |  pamsumm -mean | sed -e 's/^[^0-9]*//'`
    done
    i=`echo $1 | awk '{printf("%8d",$0)}'`
    echo "$i" $C0 $C1 $C2
}

ADCport=${ADCport:-"ttyACM0"} # Arduino UNO communication port in /dev/...
ADCrate=${ADCrate:-115200}    # communication baud rate
ADChw=${ADChw:-"arduino:avr:uno"} # ADC hw card (defaults to Arduino UNO)
ADCfw="adcsampler"                # file name for ADC firmware .ino source

# see https://stackoverflow.com/questions/3918032/bash-serial-i-o-and-arduino
# talks to Arduino UNO running adcsampler.ino and prints measurements to stdout
# DS 20190124, see talk.sh
function sample_ADC()
{
 if [ -r /dev/$ADCport ]; then
# set up baud rate etc, this seems to be reasonable:
    stty -F /dev/$ADCport $ADCrate cs8 cread clocal sane

# however, the "sane" option screws smth bad
# so when all else fails, check settings after running arduino IDE
# (see https://playground.arduino.cc/Interfacing/LinuxTTY)
# stty -a < /dev/ttyACM0
# speed 115200 baud; rows 0; columns 0; line = 0;
# intr = ^C; quit = ^\; erase = ^?; kill = ^H; eof = ^D; eol = <undef>;
# eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
# werase = ^W; lnext = ^V; flush = ^O; min = 0; time = 0;
# -parenb -parodd cs8 -hupcl -cstopb cread clocal -crtscts
# -ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr -icrnl -ixon -ixoff
# -iuclc -ixany -imaxbel -iutf8
# -opost -olcuc -ocrnl -onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
# -isig -icanon -iexten -echo -echoe -echok -echonl -noflsh -xcase -tostop -echoprt
# -echoctl -echoke

    stty -F /dev/$ADCport $ADCrate cs8 cread clocal  \
  -parenb -parodd -hupcl -cstopb -crtscts \
  -ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr -icrnl \
```

```
  -ixon -ixoff -iuclc -ixany -imaxbel -iutf8 \
  -opost -olcuc -ocrnl -onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0 \
  -isig -icanon -iexten -echo -echoe -echok -echonl -noflsh -xcase -tostop -echoprt \
  -echoctl -echoke

# Arduino will reset when the serial device is opened for the first time,
# but will not reset when the serial device is opened again
# open port by redirecting to file description 3
    exec 3<> /dev/$ADCport
# wait for Arduino's initialization (at least 2 sec ...)
    sleep 2
# communicate to Arduino, set up sampling rate and start measurements
    echo -n "T"${1:-10000}"N"${2:-5}"HS" >&3
# cat  <&3
# -Wposix is required to read hexadecimal data in, see
# https://stackoverflow.com/questions/4614775/converting-hex-to-decimal-in-awk-or-sed
# use fflush(stdout) to see smth with tail -f during slow data reception
# (may not work on older versions of awk)
# count the number of (hexadecimal->integer) output data lines and stop
    awk -Wposix -v "npts=${2:-5}" 'BEGIN{cnt=0} /^#/{print $0; next;} {val=("0x" $1)+0; printf("%d %4d\n",cnt++,
        val);} cnt>=npts {exit 0;} END{print "#EOF"}' <&3
# NB repeated calls to Arduino go without delays since they do not initialize Arduino
#   echo -n "S" >&3; awk -Wposix -v ...

#  close Arduino communication port
    exec 3>&-
 else
     echo "#"$OURPROMPT"Arduino serial port $ADCport is inacessible!"
 fi
}
# ------------------------------------------------------------------------------
# the simple solution based on arduino is suggested in
# https://arduino.stackexchange.com/questions/15893/how-to-compile-upload-and-monitor-via-the-linux-command-line
# an alternative uses arduino-builder from arduino IDE distro for compilation
# see https://github.com/pavelmc/amake
#   amake -u uno adcsampler.ino /dev/$ADCport
# see also http://playground.arduino.cc/Learning/CommandLine
#          https://github.com/arduino/arduino-cli
# ------------------------------------------------------------------------------
case $1 in
    'loadADC') # upload the fw code using official arduino cli TODO: test this
  echo "${OURPROMPT} upload $ADCfw.ino firmware to Arduino UNO on /dev/$ADCport"
        # https://github.com/arduino/Arduino/blob/master/build/shared/manpage.adoc
  arduino --upload $ADCfw.ino --port /dev/$ADCport --board $ADChw
  ;; # or should one descend to ~/Arduino/$ADCfw/$ADCfw.ino ??
    'sampleADC')
  sample_ADC $ADCTIME $ADCNPTS > $adcdata
  ;;
    'testADC')
  sample_ADC
  ;;
    'ADCplot') # use nmin nmax (or nfft ?)
  cat <<EOF | gnuplot
#!/usr/bin/gnuplot
set encoding iso_8859_1 # to have 1/2 in Times
set terminal postscript landscape enhanced color solid "Helvetica" 16
set macros  # required for @-expansions to work (gnuplot >=4.2) ...
dataf=sprintf('"%s" u (tval(\$1)):(uval(\$2))',"$adcdata");
set output "$adcplotf";
print "${OURPROMPT}gnuplot: ",dataf," -> ","$adcplotf"
# extract sampling interval T, number of samples N, and sampling rate F
`awk '/period/ {gsub(/[^0-9. ]*/,""); print "T=" $1 ";", "N=" $2 ";", "F=" $3}' $adcdata`

nmin=$nmin;             # start of the displayed data fragment, use 0..N-1
if (nmin >= N || nmin<0) nmin=0;
nmax=nmin+2**$nfft;     # end of the displayed data fragment, normally N=$ADCNPTS
if (nmax > N || nmax-nmin<=1) nmax=N-1;
adcT=T/1000.0;          # value of sampling period in msec ($ADCTIME/1000)
tval(n) = n*adcT;       # value of sample time in msec
uval(v) = v/1023.0*5;   # measured potential in volts
Tmin=tval(nmin)-adcT/2; # displayed fragment begin
Tmax=tval(nmax)+adcT/2; # displayed fragment end
print "${OURPROMPT}gnuplot: T=",T/1000," msec, N=",N," samples at ",F," Hz for ", Tmax/1000," sec"
print "${OURPROMPT}gnuplot: displayed data nmin=",nmin,", nmax=",nmax," t=",int(Tmin),"..",int(Tmax)," msec"
set grid
set xrange [Tmin:Tmax]; set xlabel "Time [msec]"
```

```
set yrange [0:5];  set ylabel "Arduino UNO ADC pin A0 [V]"

set label 101 sprintf("data %i-%i at %g %sHz",nmin,nmax,(F>=1000?F/1000.0:F),(F>=1000?"k":"")) \
          at graph -0.08,-0.09 tc rgb 'blue'

set label  99 sprintf("{/=12 ver. $VERSION, `date +%Y-%m-%d`  {/Symbol \323}`date +%Y` Sadovskií}") \
          at graph 1.0125,0.005 rotate front tc rgb 'gray'

set label  98 sprintf("{/=12 $adcdata: %i samples with %g {/Symbol m}sec interval",N,T) at graph 0.005,1.025
    front tc rgb 'gray'

plot @dataf not w l lt 3 lw 2
#     @dataf t 'V_{out}' w p pt 7 ps 0.85 lt 3 lw 1
EOF
  ps2pdf $adcplotf && echo $OURPROMPT"created `basename $adcplotf .ps`.pdf"
  ;;
    'FTADCplot')
  if ! [ -r "$adcdata" ]; then
      echo $OURPROMPT" file $adcdata not found, bailing out ..."
      exit 1;
  fi
  csvbase=$(basename "$adcdata" .dat)
  echo $OURPROMPT"file $adcdata";
# extract sampling interval T, number of samples N, and sampling rate F
  eval `awk '/period/ {gsub(/[^0-9. ]*/,""); print "adcT=" $1 ";", "adcN=" $2 ";", "adcF=" $3}' $adcdata`
  echo $OURPROMPT"2^$nfft points, starting at $nmin with sampling interval ${adcT} usec";
# some useful hints on maxima i/o can be found here
# http://cs.swan.ac.uk/~csoliver/ok-sat-library/internet_html/doc/doc/Maxima/BookWoollett/mbe1start.pdf
  cat <<EOF | maxima --very-quiet > /dev/null
fpprintprec:4$
load(fft)$    /* see maxima fft example from the manual */
n : 2^$nfft $    /* number of points in the sample */
x : make_array(flonum, n) $
a : make_array(flonum, n/2+1) $
b : make_array(flonum, n/2+1) $
fillarray(x,
`awk 'NR==1 {j=0;i=1; nmax=2^nfft} /^[^0-9]/ || ++j<jmin {next;} i>nmax {exit;} {print (i-1?" ":"[") $2 (i<nmax?
    ",\\\\":"]"); i++ }' jmin=$nmin nfft=$nfft "$adcdata"`) $
y : fft(x) $
a[ 0 ] : realpart(y[0]) $
a[n/2] : realpart(y[n/2]) $
b[n/2] : b[0] : 0 $
for k : 1 thru n/2-1 do
  ( a[k] : realpart(y[k]+y[n-k]),
    b[k] : imagpart(y[n-k]-y[k])
  );
done $
with_stdout( "FT-${csvbase}.dat",
   for k : 0 thru n/2 do printf(true,"~5d ~8,5f ~8,5f  ~10,5f~%", k, a[k], b[k], x[k] ) ) $
EOF
  maxFTadc=`awk 'BEGIN {x=0; i=0} i++<1 || (x=$2^2+$3^2)>a {a=x} END {print sqrt(a)}' "FT-${csvbase}.dat"`
  echo $OURPROMPT"FT spectrum in FT-${csvbase}.dat with max AC amplitude $maxFTadc";
# upper frequency cutoff, defaults to the whole range ...
  plotfmax=${plotfmax:-`awk 'END {print 2^nfft/2-1}' nfft=$nfft < /dev/null`}

#  f(j) := sum (a[k]*cos(2*%pi*j*k/n) + b[k]*sin(2*%pi*j*k/n), k, 0, n/2) $
#  makelist (float (f (j)), j, 0, n - 1);
#  so for index k, the frequency is k/n (the period is n/k)
# ------------------------------------------------------------------------------
  cat <<EOF | gnuplot
# plot FT's of ADC data (c)2019 D.Sadovskii
set terminal postscript landscape enhanced color solid "Helvetica" 16
set encoding iso_8859_1
set output "FT$adcplotf"
A(a,b)=sqrt(a**2+b**2); # amplitude of a*cos(t)+b*sin(t)
uval(v) = v/1023.0*5;   # measured potential in volts
fps=$adcF; # sampling rate
Amax=$maxFTadc
fmax=$plotfmax
Vmax=uval(Amax); # max ac amplitude in V
v_DC=uval(`awk 'BEGIN {i=0} i {exit;} /^#/ {next;} {print \$2; i++}' $adcdata`);

set ytics offset 0.75,0
set xtics out nomirror offset 0,0.4
set mxtics
set x2tics out offset 0,-0.45
```

```
set mx2tics
set grid x2tics

set label 97 "{/=10 %g échantillons commençant par $nmin}",(2**$nfft) at graph 0,-0.12 left tc rgb "blue"
set label 95 "{/=10 fps=%g Hz}",fps at graph 0,-0.086 front left tc rgb "blue"
set label 94 sprintf("{/=10 %.3fV DC}",v_DC) at graph -0.02,0.04 rotate front left tc rgb "blue"
set label 98 "{/=8 donnés in $adcdata (at %g Hz) -> FT-${csvbase}.dat}",fps at graph 1.008,0.0 left rotate
set label 99 "{/=8 ${OURPROMPT}`basename FT$adcplotf .ps` `date +%Y-%m-%d`}" at graph 0,1.12 left tc lt 9
set label 96 sprintf("{/=8 ver. $VERSION  {/Symbol \323}`date +%Y` Sadovskií}") \
          at graph 1.0125,1.12 front right tc rgb 'gray'

set yrange [0:1]
set ylabel sprintf("{/=18 Amplitude relative à a_{max}=%.3f V ac}",Vmax) offset 2,0
set xrange [0:fmax]
set xlabel "{/=18 Fréquence * 2^{$nfft} / fps}" offset 0,0.25
set x2range [0:fmax*fps/(2**$nfft)]
set x2label "{/=18 Fréquence [Hz]}" offset 0,-0.2

print "${OURPROMPT}gnuplot: FT-${csvbase}.dat plotted in file FT$plotf"
set bars small
plot "FT-${csvbase}.dat" u 1:(0):(0):(A(\$2,\$3)/Amax) \
     not with yerrorbars lt 3 lw 3 pt 0
EOF
  ps2pdf FT$adcplotf && echo $OURPROMPT"converted FT$adcplotf -> `basename FT$adcplotf .ps`.pdf"
  ;;
    'frames')
  echo $OURPROMPT"Extract all frames from the movie (be patient and have disk space)"
  mplayer -quiet -nosound -benchmark -vo "jpeg:outdir=$wrkdir" $movie
  frame_info
  # preserve information on the extracted frames
  frame_info > README.frames
  ID_FRAME_DATE=`date +%Y%m%d-%H%M%S`
  echo "ID_FRAME_DATE=$ID_FRAME_DATE" >> README.frames
  ;;
    'sample')
  echo $OURPROMPT"Work on one particular sample frame $sample_frame"
  echo $OURPROMPT"crop to a ${crop_XX}x${crop_YY} rectangle sample starting at (${crop_X0},${crop_Y0}), see
      $sample_file"
  # jpegtopnm `set_ffname $sample_frame` | pamcut  $crop_X0 $crop_Y0 $crop_XX $crop_YY | pnmsmooth >
      $sample_file
  # cat $sample_file | pamchannel 0 |  pamsumm -mean
  # cat $sample_file | pamchannel 1 |  pamsumm -mean
  # cat $sample_file | pamchannel 2 |  pamsumm -mean
  echo "#  frame  R         G         B"
              #   9498 77.197059 26.172549 25.048529
  frame_RGB $sample_frame
  ;;
    'histo')
  fmax=`frame_count`;
  echo "#  frame  R         G         B"
  for (( f=1 ; f<=fmax ; f++ )) ; do
      frame_RGB $f
      (( f%100==0 )) && echo "   analyzed $f frames of $fmax" > /dev/stderr
  done
  ;;
    'plot')
  cat <<EOF | gnuplot
# plot R,G,B channel variations (c)2013 D.Sadovskii
set terminal postscript landscape enhanced color solid "Helvetica" 16
set output "$plotf"

# the fps=${ID_VIDEO_FPS} value supplied by mplayer is bogous
# compute the real frame per second rate using length=${ID_LENGTH}
fps=`frame_count`/(${ID_LENGTH})
print "${OURPROMPT}gnuplot: fps=", fps, " ${ID_LENGTH}sec - ${ID_START_TIME}sec (supplied fps=$ID_VIDEO_FPS)";
# hmm or should it be like
# fps=`frame_count`/(${ID_LENGTH}-${ID_START_TIME})

set label  98 "{/=8 from {$movief} [$dataf] $ID_VIDEO_FORMAT ${ID_VIDEO_WIDTH}x${ID_VIDEO_HEIGHT} ${ID_LENGTH}
    sec, `frame_count` frames (%gfps)}",fps at graph 1.012,0.0 left rotate tc lt 9
set label  99 "{/=8 $OURPROMPT`basename $plotf .ps` {/Symbol \323}2013 DS}" at graph 0,1.025 left tc lt 9

set xlabel "{/=18 Video frame}" offset 0,0.25
set ylabel "{/=18 Averaged per channel intensity}" offset 1.5,0
set mxtics
```

```
plot \
     "$dataf" using 1:2 title "rouge (R)" w l lt 1 lw 2, \
     "$dataf" using 1:3 title "vert (G)" w l lt 2 lw 2, \
     "$dataf" using 1:4 title "bleu (B)" w l lt 3 lw 2

EOF
  ps2pdf $plotf && echo $OURPROMPT"created `basename $plotf .ps`.pdf"
  ;;
    "FTplot")
  for nchn in `seq 3`; do
      echo $OURPROMPT"fast Fourier transform on channel $nchn"
      cat <<EOF | maxima --very-quiet | awk '/^ *[1-9][0-9]* / {print $0}' > FT${nchn}${dataf}
fpprintprec:4$
load(fft)$    /* see maxima fft example from the manual */
n : 2^$nfft $    /* number of points in the sample */
x : make_array(flonum, n) $
a : make_array(flonum, n/2+1) $
b : make_array(flonum, n/2+1) $
fillarray(x,
`awk 'NR==1 {j=0;i=1; nmax=2^nfft} /^ *#/ || j++<jmin {next;} i>nmax {exit;} {print (i-1?" ":"[") $(chan+1) (i<
    nmax?",\\\\":"]"); i++ }' jmin=$nmin nfft=$nfft chan=$nchn $dataf`) $
y : fft(x) $
a[ 0 ] : realpart(y[0]) $
a[n/2] : realpart(y[n/2]) $
b[n/2] : b[0] : 0 $
for k : 1 thru n/2-1 do
  ( a[k] : realpart(y[k]+y[n-k]),
    b[k] : imagpart(y[n-k]-y[k])
  );
done $
for k : 0 thru n/2 do
  ( printf(true,"~5d ~8,5f ~8,5f~%", k, a[k], b[k]) );
done $
EOF
      eval maxFT${nchn}=`awk 'BEGIN {a=0} (x=$2^2+$3^2)>a {a=x} END {print sqrt(a)}' FT${nchn}${dataf}`
  done
# maxima --very-quiet -r 'batchload("prog.maxi")$'
#  f(j) := sum (a[k]*cos(2*%pi*j*k/n) + b[k]*sin(2*%pi*j*k/n), k, 0, n/2) $
#  makelist (float (f (j)), j, 0, n - 1);
#  so for index k, the frequency is k/n (the period is n/k)
# ------------------------------------------------------------------------------
  cat <<EOF | gnuplot
# plot FT's of R,G,B channel variations (c)2013 D.Sadovskii
set terminal postscript landscape enhanced color solid "Helvetica" 16
set encoding iso_8859_1
set output "FT$plotf"
A(a,b)=sqrt(a**2+b**2); # amplitude of a*cos(t)+b*sin(t)
# compute the real frame per second rate
fps=`frame_count`/(${ID_LENGTH:-1}) # ID_LENGTH must be set, if not it is a bug!
fmax=$plotfmax
set ytics offset 0.75,0
set xtics out nomirror offset 0,0.4
set mxtics
set x2tics out offset 0,-0.45
set mx2tics
print "Original movie file $gpmovief, data file $dataf"
set label 97 "{/=10 %g TF points, commençant par $nmin}",(2**$nfft) at graph 0,-0.12 left tc lt 9
set label 98 "{/=8 vidéo {$gpmovief} [FT $dataf] $ID_VIDEO_FORMAT ${ID_VIDEO_WIDTH}x${ID_VIDEO_HEIGHT} ${
    ID_LENGTH}sec, `frame_count` cadres (%gfps)}",fps at graph 1.015,0.0 left rotate
set label 99 "{/=8 ${OURPROMPT}`basename FT$plotf .ps` {/Symbol \323}2013 DS}" at graph 0,1.12 left tc lt 9

set yrange [0:*]
set ylabel "{/=18 Intensité par couleur}" offset 2,0
set xrange [$nmin:$nmin+(2**$nfft)-1]
set xlabel "{/=18 Cadre }" offset 0,0.25
set x2range [$nmin/fps:($nmin+(2**$nfft)-1)/fps]
set x2label "{/=18 Temps [sec]}" offset 0,-0.2

plot \
     "$dataf" using 1:2 title "rouge (R)" w l lt 1 lw 2, \
     "$dataf" using 1:3 title "vert (G)" w l lt 2 lw 2, \
     "$dataf" using 1:4 title "bleu (B)" w l lt 3 lw 2

set yrange [0:1]
set ylabel "{/=18 Amplitude relative}" offset 2,0
set xrange [0:fmax]
```

```
set xlabel "{/=18 Fréquence * 2^{$nfft} / fps}" offset 0,0.25
set x2range [0:fmax*fps/(2**$nfft)]
set x2label "{/=18 Fréquence [Hz]}" offset 0,-0.2

print "FT plots in file FT$plotf"
set bars small
plot \
  "FT1${dataf}" u 1:(0):(0):(A(\$2,\$3)/$maxFT1) t "R_{max}=$maxFT1" with yerrorbars lt 1 lw 2 pt 0
plot \
  "FT2${dataf}" u 1:(0):(0):(A(\$2,\$3)/$maxFT2) t "G_{max}=$maxFT2" with yerrorbars lt 2 lw 2 pt 0
plot \
  "FT3${dataf}" u 1:(0):(0):(A(\$2,\$3)/$maxFT3) t "B_{max}=$maxFT3" with yerrorbars lt 3 lw 2 pt 0

EOF
  ps2pdf FT$plotf && echo $OURPROMPT"created `basename FT$plotf .ps`.pdf"
  ;;


    # plotting data from csv files produced by Pierre
    "csvplot")
  if ! [ -r "$csvfile" ]; then
      echo $OURPROMPT" file $csvfile not found, bailing out ..."
      exit 1;
  fi
  csvbase=$(basename "$csvfile" .csv)
  echo $OURPROMPT"file $csvfile";
  csvtstep=`awk 'BEGIN{FS=";"} NR==1 {j=0;i=1; nmax=2^nfft} /^[^0-9]/ || ++j<jmin {next;} i>nmax {exit;} {gsub
      (/,/,".");  t=$1-(i==1?t0=$1:t0); i++} END {print t/(nmax-1)}' jmin=1 nfft=2 "$csvfile"`
  echo $OURPROMPT"2^$nfft points, starting at $nmin with estimated time step of ${csvtstep} sec";
  cat <<EOF  | maxima --very-quiet | awk '/^ *[1-9][0-9]* / {print $0}' > "FT-${csvbase}.dat"
fpprintprec:4$
load(fft)$    /* see maxima fft example from the manual */
n : 2^$nfft $    /* number of points in the sample */
x : make_array(flonum, n) $
a : make_array(flonum, n/2+1) $
b : make_array(flonum, n/2+1) $
fillarray(x,
`awk 'BEGIN{FS=";"} NR==1 {j=0;i=1; nmax=2^nfft} /^[^0-9]/ || ++j<jmin {next;} i>nmax {exit;} {gsub(/,/,".");
    print (i-1?" ":"[") $2 (i<nmax?",\\\\":"]"); i++ }' jmin=$nmin nfft=$nfft "$csvfile"`) $
y : fft(x) $
a[ 0 ] : realpart(y[0]) $
a[n/2] : realpart(y[n/2]) $
b[n/2] : b[0] : 0 $
for k : 1 thru n/2-1 do
  ( a[k] : realpart(y[k]+y[n-k]),
    b[k] : imagpart(y[n-k]-y[k])
  );
done $
for k : 0 thru n/2 do
  ( printf(true,"~5d ~8,5f ~8,5f~%", k, a[k], b[k]) );
done $
EOF
  maxFTcsv=`awk 'BEGIN {a=0} (x=$2^2+$3^2)>a {a=x} END {print sqrt(a)}' "FT-${csvbase}.dat"`
  echo $OURPROMPT"maximal FT spectrum amplitude is $maxFTcsv";
  ln -sf "$csvfile" csvfile.csv
  cat <<EOF | gnuplot
# FT spectrum of the csv file (c)`date +%Y` D.Sadovskii
set terminal postscript landscape enhanced color solid "Helvetica" 16
set encoding iso_8859_1
set output "csvplot.ps"
print "${OURPROMPT}original csv ${csvbase}, plots in csvplot.ps"
A(a,b)=sqrt(a**2+b**2); # amplitude of a*cos(t)+b*sin(t)
# compute the real frame per second rate
fps=1.0/${csvtstep};
nmax=`awk 'BEGIN{j=0} /^[0-9]/ {j++} END{print j}' csvfile.csv`
print fps,nmax
set label 97 "{/=10 %g TF points, commençant par $nmin}",(2**$nfft) at graph 0,-0.12 left tc lt 9
set label 98 "{/=8 ${csvbase}, %gHz}",fps,"{/=8 { } %g pts}",nmax at graph 1.015,0.0 left rotate
set label 99 "{/=8 ${OURPROMPT} {/Symbol \323}2013 DS}" at graph 0,1.12 left tc lt 9
set ytics offset 0.75,0
set mxtics
set xtics out nomirror offset 0,0.4
set xrange [0:nmax]
set x2range [0:(nmax-1)*$csvtstep]
set x2label "{/=18 Temps [sec]}" offset 0,-0.2
set x2tics out offset 0,-0.45
```

```
set mx2tics

set ylabel "{/=18 Amplitude}" offset 2,0
set xlabel "{/=18 Cadre (per ${csvtstep} sec)}" offset 0,0.25
plot \
  "< awk -F';' '/^[0-9]/ {gsub(/,/,\\".\\"); print \$1,\$2}' csvfile.csv" u 0:2 t "csv data" with l lt 3 lw 2

set xrange [$nmin:$nmin+(2**$nfft)]
set x2range [($nmin-1)*$csvtstep:($nmin+(2**$nfft)-1)*$csvtstep]
plot \
  "< awk -F';' '/^[0-9]/ {gsub(/,/,\\".\\"); print \$1,\$2}' csvfile.csv" u 0:2 t "csv data" with l lt 3 lw 2


fmax=$plotfmax
set yrange [0:1]
set ylabel "{/=18 Amplitude relative}" offset 2,0
set xrange [0:fmax]
set xlabel "{/=18 Fréquence * 2^{$nfft} / fps}" offset 0,0.25
set x2range [0:fmax*fps/(2**$nfft)]
set x2label "{/=18 Fréquence [Hz]}" offset 0,-0.2
set x2tics out offset 0,-0.45
set mx2tics
set bars small

plot \
  "FT-${csvbase}.dat" u 1:(0):(0):(A(\$2,\$3)/$maxFTcsv) t "B_{max}=$maxFTcsv" with yerrorbars lt 3 lw 2 pt 0

EOF
  ps2pdf csvplot.ps
  ;;

# http://www.scholarpedia.org/article/Oregonator
# model chemical kinetics using standard Maxima RK4 o.d.e. solver
    "model")
  RK_TFIN=${RK_TFIN:-30};
  RK_STEP=${RK_STEP:-"0.01"};
  RK_Fval=${RK_Fval:-1};
  RK_Aval=${RK_Aval:-"0.06"};
  RK_Bval=${RK_Bval:-"0.02"};
  cat <<EOF  | maxima --very-quiet
load("./rkf45.mac");
log10(x) := log(x) / log(10)$
/* rate constants of known chemical reactions with [H+]=0.8M */
k1 : 1.28$    /* 1/(M.sec) */
k2 : 2.4e6$
k3 : 33.6$
k4 : 2.4e3$
kc : 1$
cH : 0.8$      /* invariant initial concentration of [H+]          */
B : $RK_Bval$  /* invariant initial concentration of CH2(COOH)2    */
A : $RK_Aval$  /* invariant initial concentration of BrO3-         */
f : $RK_Fval$  /* 0.5..2.4 adjustable stoichiometric parameter     */

 ode : [ k1*A*Y - k2*X*Y + k3*A*X - 2*k4*X^2,  /* dX/dt */
        -k1*A*Y - k2*X*Y + kc/2*f*B*Z,         /* dY/dt */
         2*k3*A*X - kc*B*Z ]$                  /* dZ/dt */

/* scaled version known as the Field-Noyes equations */
svars: [x=2*k4*X/(k3*A), y=k2*Y/(k3*A), z=kc*k4*B*Z/(k3*A)^2, tau=kc*B*t]$
 vars: float(solve(svars,[X,Y,Z,t]))$
xyz0 : [1,1,1]$
XYZ0 : subst([x=1,y=1,z=1], subst(vars,[X,Y,Z]))$

eps  : kc*B/(k3*A)$
eps1 : 2*kc*k4*B/(k2*k3*A)$
q    : 2*k1*k4/(k2*k3)$
sode : [ ( q*y - x*y + x*(1-x))/eps,
         (-q*y - x*y + f*z    )/eps1, x-z ]$

/* with_stdout( "model-${csvbase}.dat" */
/*
 sol : rk( ode, [X, Y, Z], XYZ0, [  t, 0, $RK_TFIN, $RK_STEP])$

print("# [X, Y, Z] give concentrations of HBrO2 , Br-, and cerium-IV Ce4+")$
print("#  time      X       Y       Z")$
for s in sol do
```

```
  printf(true,"~9,4f  ~6,4f  ~6,4f  ~6,4f~%", s[1], s[2], s[3], s[4]) $
done$
*/
/*
ssol : rk(sode, [x, y, z], xyz0, [tau, 0, 0.1, $RK_STEP/200])$
*/
ssol : rkf45(sode, [x, y, z], xyz0, [tau, 0, 0.1]);

printf(true,"# scaled parameters eps=~,2e, eps1=~,2e, q=~,2e, and f=~f ~%",
       eps,eps1,q,f)$
print("# [x, y, z] give scaled concentrations of HBrO2 , Br-, and cerium-IV Ce4+")$
print("#  time      x       y       z")$
for s in ssol do
  printf(true,"~9,4f  ~6,4f  ~6,4f  ~6,4f~%", s[1], log10(abs(s[2])), log10(abs(s[3])), log10(abs(s[4]))) $
done$

/*
test : rk([v,-u],[u,v],[1,0],[t,0,2*%pi,0.05*%pi])$
print("#  time      x       y")$
for s in test do
  printf(true,"~9,4f  ~6,4f  ~6,4f  ~6,4f ~%", s[1]/%pi, s[2], s[3], cos(s[1]) ) $
done$
*/

EOF
  ;;

    "record")
  mencoder -tv driver=v4l2:noaudio:width=640:height=480:outfmt=mjpeg:fps=$RECFPS:brightness=-10 \
      tv:// -endpos 00:00:30 -vf scale -fps $RECFPS -ofps $RECFPS \
      -o $movie \
      -ovc lavc -lavcopts vcodec=mpeg4 -nosound -ffourcc xvid
#  HERE using -vf scale seems to shorten some of the unknown colorspace complaints
#  THIS SETS UPPER TIME LIMIT (DURATION), -ss 00:00:02 CAN SET START TIME
#  USING -of mpeg PRODUCES AN MPEG FILE but some how of mediocre color quality ?
  ;;
     "video")
  echo "webcam test: press q (in the video screen) to exit"
  mplayer tv://
  ;;
    "ver")
  echo $VERSION
  ;;
    "help"|"?")
  cat <<EOF
 $OURPROMPT Linux/MacOS bash script for oscillating reaction lab work
 (c) `date +%Y` DS <sadovski at univ-littoral.fr> rev. $VERSION
------------------------------------------------------------------------------
 requires
    mplayer and netpbm for video sequence treatment
    maxima for Fourier transforms, and gnuplot for all graphics

 To determine the area of color sampling (should be done once) use any
 image viewer that allows cropping such as the old xv on linux/unix,
 or do a series of trial crops using netpbm's routines ...

 for basic usage call

    <settings> mkhisr.sh operation

 where "settings" is a space separated variable=value list of required variables
 and operation is one of the following

    help   see this message
     ver   display script's current version [$VERSION]
   video   webcam alignment test: press q (in the video screen) to exit
  record   record the movie to the file $movie
  frames   extract all frames from the video file $movief
  sample   crop a rectangle from sample frame $sample_frame to file $sample_file
   histo   produce a list of R,G,B data from extracted files, store in $dataf
    plot   represent R,G,B data graphically in file `basename $plotf .ps`.pdf
 FTplot   compute and plot Fourier transform (FT) of the data in $dataf
csvplot   plot legacy format data (such as the ones from Orphy GTI)
loadADC   upload the fw code $ADCfw.ino using official Arduino cli
testADC   sample test (a few points) with $ADChw on /dev/$ADCport
sampleADC  record data from Arduino ADC to file $fnbase-ADC.dat
```

```
 ADCplot   plot data obtained from Arduino ADC to file $adcplotf
FTADCplot  plot FT of the ADC data in `basename $adcdata` to `basename FT$adcplotf`.pdf
    model  model chemical kinetics using standard Maxima RK4 o.d.e. solver

 variables that can/should appear in the settings and their [default values]

  RECFPS   video frames per sec        [$RECFPS]
  ADCTIME  ADC sampling period in usec [${ADCTIME}]
  ADCNPTS  total number of ADC samples [${ADCNPTS}]
  crop_X0 crop_Y0 position of sample rectangle [$crop_X0,$crop_Y0]
  crop_XX crop_YY size of sample rectangle     [$crop_XX,$crop_YY]
  sample_frame    frame to crop the test sample from [${sample_frame}]
  fnbase   prefix to construct unique file names [$fnbase]
  wrkdir   working folder [${wrkdir}]
  srcdir   folder with the movie and ADC data [$srcdir]
  movief   file name with the movie [$movief]
  csvfile  file with external csv data [${csvfile}]
  nmin     starting point (frame) for the FT sample interval [${nmin}]
  nfft     2^nfft is the length of the FFT sample interval   [${nfft}]
  plotfmax upper cutoff in the FT frequency spectrum [${plotfmax}]

 for example, call (in this sequence):
    mkhisr.sh frames
    crop_X0=346 crop_Y0=188 crop_XX=24 crop_YY=85 mkhisr.sh sample
    mkhisr.sh histo > RGB.dat

 for legacy csv data files
    nmin=400 nfft=10 mkhisr.sh csvplot
 ------------------------------------------------------------------------------
 people behind the development of this project:
 Dmitrii Sadovskií     initiateur du projet, informatique, Calais
 Anton Sokolov         intervenant TP L3, lab setup, Dk
 Pierre Kulinski       realisateur partie mesures, Dk
 Amaury Kasprowiak     realisateur partie chimie, Dk

EOF
  frame_info
  ;;
    *)
  frame_info
esac
```

B Code ARDUINO

Le programme `adcsampler.ino` fait partie de TP ARDUINO, sec. 2.1.6. Avec la connexion usb/série à la vitesse de 115200 baud, ce programme assure le taux d'échantillonnage inférieur ou égal à 2.5kHz, largement suffisant pour les phénomènes que nous étudions.

```
/*
  This simple precise fixed time interval ADC sampler
  can be used for periods greater than 300 usec (see code, safer 500 usec)
  and thus has maximal theoretical sampling rate of 3kHz (more like 2kHz)
  see https://playground.arduino.cc/Interfacing/LinuxTTY on interfacing
  with linux serial port, which is normally /dev/ttyACM0
  stty -F /dev/ttyACM0 cs8 115200 ignbrk -brkint -icrnl -imaxbel -opost -onlcr -isig -icanon -iexten -echo -
      echoe -echok -echoctl -echoke noflsh -ixon -crtscts
  stty -F /dev/ttyACM0 115200 cs8 cread clocal
  screen /dev/ttyACM0 115200
*/
#define PIN_ANLG A0 /* pin A0..A5 for analog 5V max 10-bit ADC entry */
unsigned long READ_PERIOD = 4000;  // 4000 us gives 250 Hz sampling rate
unsigned long lastRead=0;
unsigned int npts=0; // number of samples

void setup() {
  /* Open faster serial communication (instead of usual speed 9600bps) */
  Serial.begin(115200); // 115200 bps = 14400 bytes/sec, 70 usec/byte
  while (!Serial) {
    ; /* wait for serial port to connect (for native USB port only) */
  }
  Serial.println("# fixed time interval ADC sampler");
}

void loop() {
  static unsigned int ncnt=0;
```

```
  static char key='Q';
  static long val=0;
  static char hex_format=0;
  char r;
  while(Serial.available()) { // read settings
    r = Serial.read();
    if(isDigit(r)) {
      val*=10;
      val+= r-'0';
    }
    else {
      switch(key) {  // keys that preceed numerical values
        case 'T':    // set sampling period (usec)
          if(val) READ_PERIOD = val;
          break;
        case 'N':    // set number of samples
          if(val) npts=val;
      }
      val=0;
      key=r;
      switch(key) {  // switches
        case 'H':    // hex format toggle
          hex_format^=1;
          break;
        case 'S':    // start sampling
          ncnt=npts;
        default:
          Serial.print("# period=");
          Serial.print(READ_PERIOD);
          Serial.print(" usec, samples=");
          Serial.print(npts);
          Serial.print(" at ");
          Serial.print(1000000.0 / READ_PERIOD);
          Serial.print(" Hz");
          Serial.println();
        case 'N':
        case 'T':
        case 'Q':
          break;
      }
    }
  }
  if(ncnt) { /* on 16MHz boards the time resolution is 4 usec, overrun in approx 70 min */
    for(ncnt=0, lastRead=micros()+8; ncnt<npts; ){  /* acquire and display npts samples */
      while(micros()<lastRead);
      lastRead += READ_PERIOD;
      val=analogRead(PIN_ANLG);  // 0..5V->0..1024 takes about 100 usec (10kHz)
      if(hex_format) Serial.println(val,HEX); // at least 200 usec for 3 bytes
      else Serial.println(val);
      ncnt++;
    }
    ncnt-=npts;
  }
}
```

Pour tester ce code, comme on a déjà procedé en TP ARDUINO, lancez

```
> arduino
```

Le nom du port ouvert par ARDUINO est normalement `ttyACM0`. Ouvrez le fenêtre de communication série et sélectionnez la vitesse de communication maximale de 115200 baud. Assurez vous, en (re)chargeant, si nécessaire, le programme `adcsampler.ino`, que ARDUINO répond aux commandes de genre `T2000N5S` (période 2000 μsec, 5 échantillons). En modifiant l'exposition de la photorésistance, vous devez observer les changements respectifs des valeurs affichées. Pour l'exposition maximale, elles s'approchent à 1023 (soit 5V). Vous pouvez quitter le logiciel `arduino`, et surtout *fermez son fenêtre de communication*.

C Les modalités et l'organisation de TP

L3 semestre S6, site de Calais, année 2019 Vu le nombre des étudiants et les changements au service de labo de chimie cette année, le TP «réaction oscillant» le 13/2 pour G1 et le 13/3 pour G2, 9h–12h en salle 103 sera organisé de façon suivante.

1. La préparation chimique sera entièrement à la charge des étudiants qui, étant en L3 S6, sont sensés, selon tout les opinions, d'être capables de la faire. Cette préparation sera importante pour réussir le TP car si la réaction ne démarre pas vous n'aurez rien à observer ... Vous serez notés sur votre capabilité de réaliser un projet polyvalent en physique-chimie, dont la préparation chimique fait partie.
2. Chaque séance, on fera deux préparations, qui seront étudiées dans le même appareil installé en salle TP 103 (on peut, éventuellement doubler les appareils, mais au présent cela n'est pas envisageable par manque de temps). Donc chaque groupe se partage en deux équipes, et chaque équipe est sensée de réaliser et d'étudier sa propre préparation chimique. Ainsi, au cas si une préparation est ratée, on peut se replier sur l'autre.
3. Chaque équipe choisit 1-2 personnes «en charge» de chimie. Au début de la séance, ces personnes iront au labo de chimie pour peser et préparer les solutions nécessaires, puis démarrer la réaction et la ramener en salle 103. Ainsi, chaque séance, on aura 3-4 personnes «déléguées» au labo de chimie.
4. La partie chimique est expliquée en détail dans la sec. 2 de cet énoncé. On vous propose de bien étudier cette partie ainsi que le reste de topo avant votre séance ; pour tout les questions *concrètes* concernant la préparation chimique vous pouvez vous adresser a Mme Corinne Brehier au labo de chimie.
5. L'autre partie de chaque équipe se chargera de montage, démarrage, et réglage de l'installation en salle 103 (où tout sera en pièces !). Le travail d'équipe contribue, évidemment, à votre note.