

Photopléthysmographie et santé globale

Thème : Enjeux Sociétaux

Champ : Sécurité Sanitaire – Santé globale

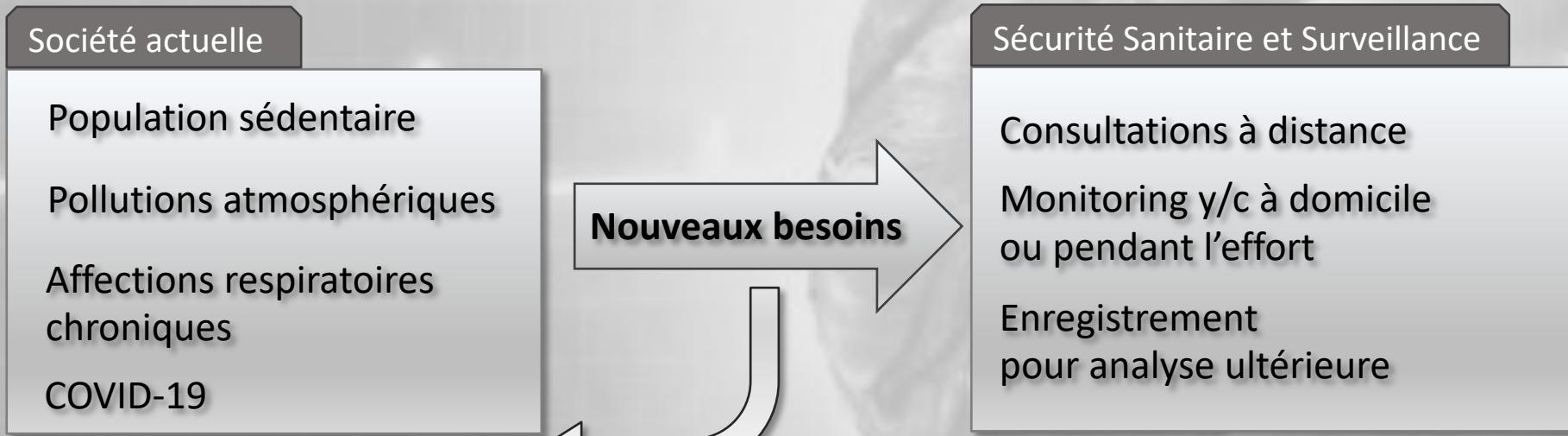


Candidat Numéro 1953

SECURITE SANITAIRE : UN ENJEU SOCIETAL PREGNANT

❖ Santé Globale et Bio-Ingénierie

- L'un des 5 enjeux sociaux identifiés par la Direction de la Recherche de l'INSA LYON
- Identification du domaine d'expertise « Instrumentation Biomédicale : Capteur, Signal, Image, Traitement »



❖ Importance de la photopléthysmographie de contact pour la sécurité sanitaire

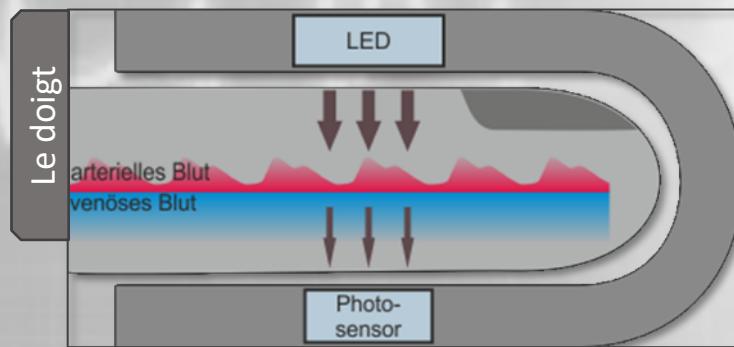
- Enregistrement permanent de la fréquence cardiaque (montres connectées)
- Enregistrement de la fréquence respiratoire ? (détection d'une affection COVID-19?)

PLAN DU DEVELOPPEMENT

- ❖ Principes de la photopléthysmographie de contact
- ❖ I - Conception et montage d'un circuit de génération et de traitement d'un signal PPG en vue d'obtenir un cardiogramme
- ❖ II - Programmation du transfert, de l'enregistrement et de l'affichage des signaux obtenus sur micro-ordinateur – essai du montage
- ❖ III - Conception et programmation d'un traitement numérique des signaux en vue d'obtenir un cardiogramme et un graphe du rythme respiratoire
- ❖ Discussions sur les résultats et ouverture

PRINCIPE DE LA PHOTOPLETHYSMOGRAPHIE (PPG)

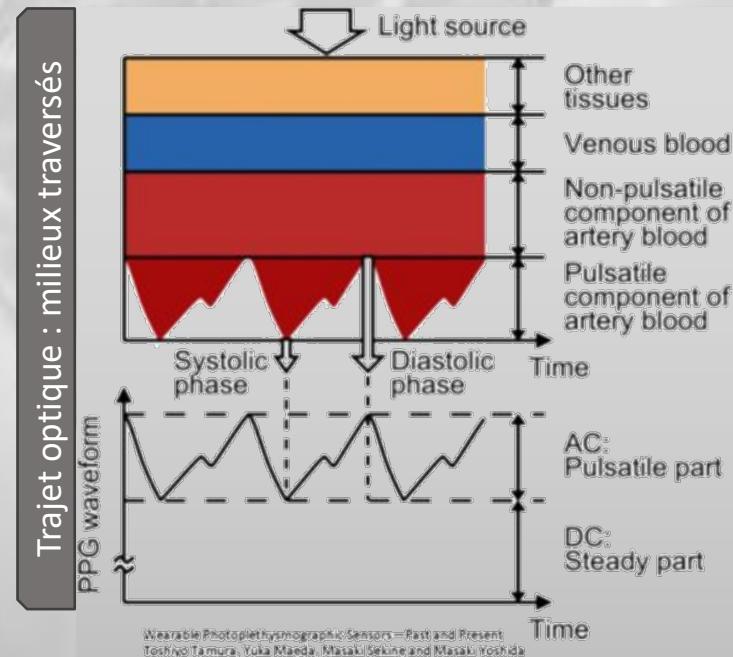
- ❖ Transformer le signal cardiaque en signal électrique en se basant sur la variabilité de l'absorption par les tissus et le sang d'une lumière émise sur la peau
- ❖ Rappel : Loi de Beer-Lambert, la solution étant ici le sang : $A_\lambda = \epsilon_\lambda \cdot I \cdot C$,
 - A_λ absorbance de la solution, sans unité, à la longueur d'onde λ
 - I est la longueur du trajet optique du rayon lumineux dans la solution, en cm,
 - ϵ_λ l'absorptivité molaire de la solution, en $L \cdot mol^{-1} \cdot cm^{-1}$. Elle dépend notamment de la longueur d'onde.



Source : <https://medis.company>

Systole:

- Variations de la longueur du trajet optique /
- Loi de Beer-Lambert avec ϵ_λ et C constants
- Variation proportionnelle à I de l'absorption par le sang
- Variation de la lumière réfléchie ou transmise à travers le doigt
- Variation d'une tension électrique (capteurs photoélectriques)



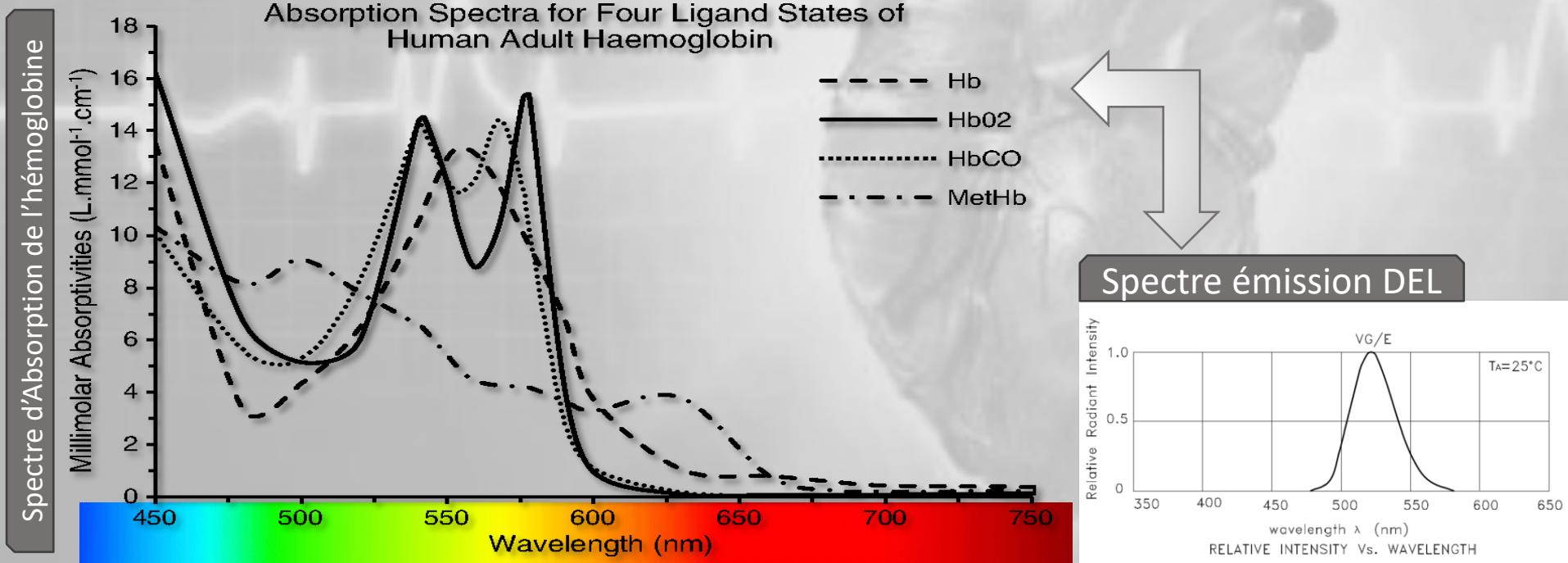
Wearable Photoplethysmographic Sensors—Past and Present
Toshiyo Tamura, Yuka Maeda, Masaki Sekine and Masaki Yoshida

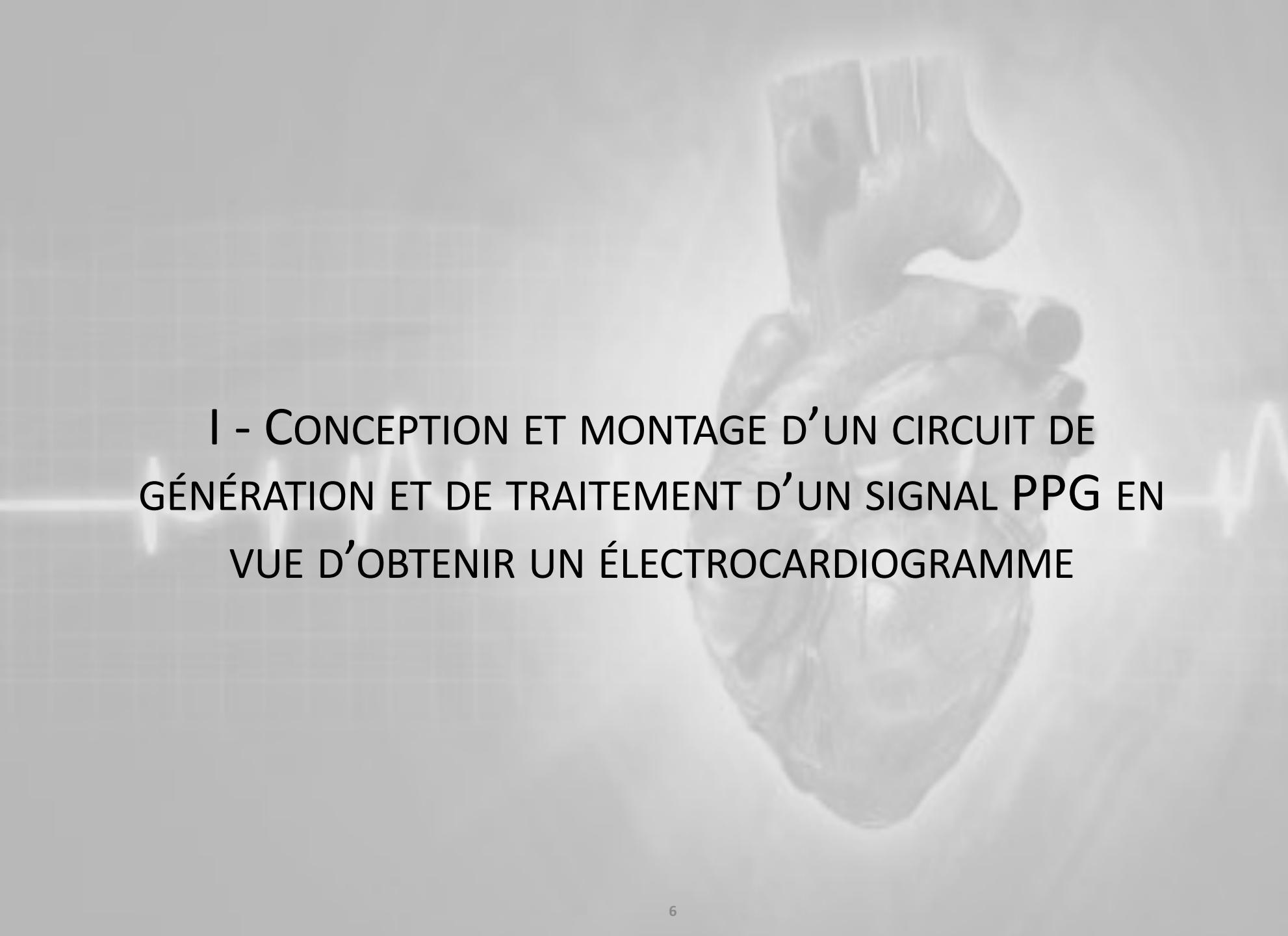
<https://www.researchgate.net> – téléchargé par Toshiyo Tamura

PRINCIPE DE LA PHOTOPLETHYSMOGRAPHIE (PPG)

❖ Longueur d'onde lumineuse à utiliser :

- ❖ Historiquement, utilisation de lumière infrarouge
- ❖ Avec l'amélioration des technologies optiques et électroniques, démocratisation de l'utilisation de lumières visibles
- ❖ => recherche d'un pic d'absorbance dans le spectre de l'hémoglobine (qui donne sa couleur au sang)



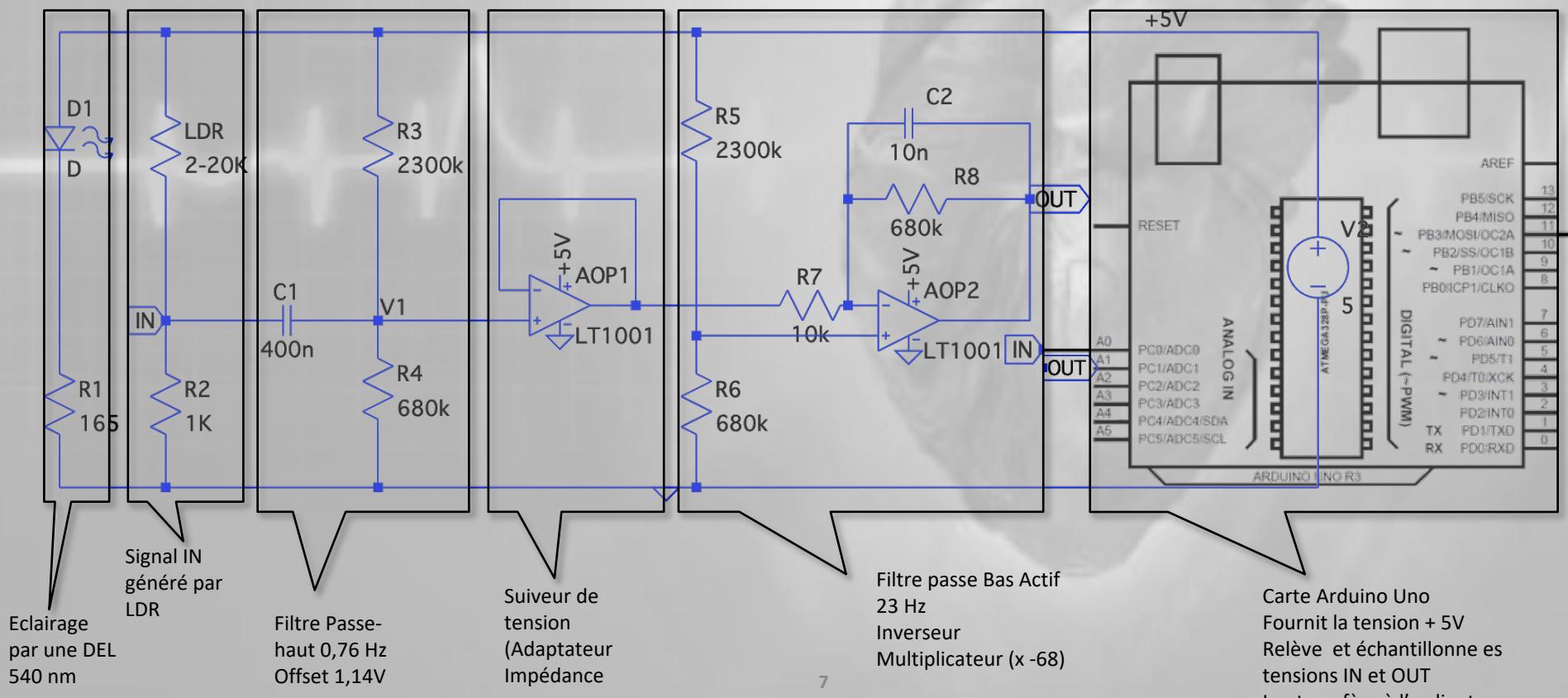


I - CONCEPTION ET MONTAGE D'UN CIRCUIT DE GÉNÉRATION ET DE TRAITEMENT D'UN SIGNAL PPG EN VUE D'OBTENIR UN ÉLECTROCARDIOGRAMME

VUE GÉNÉRALE DU CIRCUIT

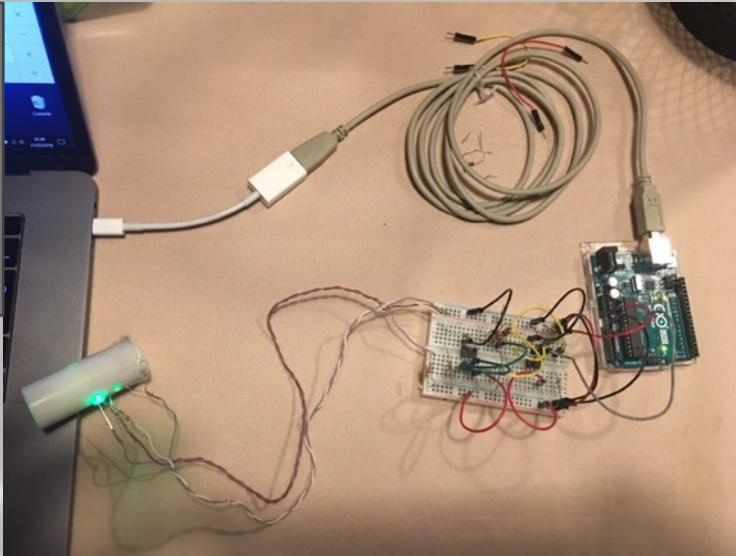
Conception avec le logiciel de simulation LTSPICE

- Master en génie Biomédical
<http://dspace.univ-tlemcen.dz/bitstream/112/10964/1/Ms.EBM.Hadj%20Ahmed%2BLalaymia.pdf>
- Article de l'entreprise Grinet
<http://grinet.blogspot.com/p/polygraph.html>
- Simplifié à l'extrême (utilisation d'une photo résistance et non pas d'une photodiode)

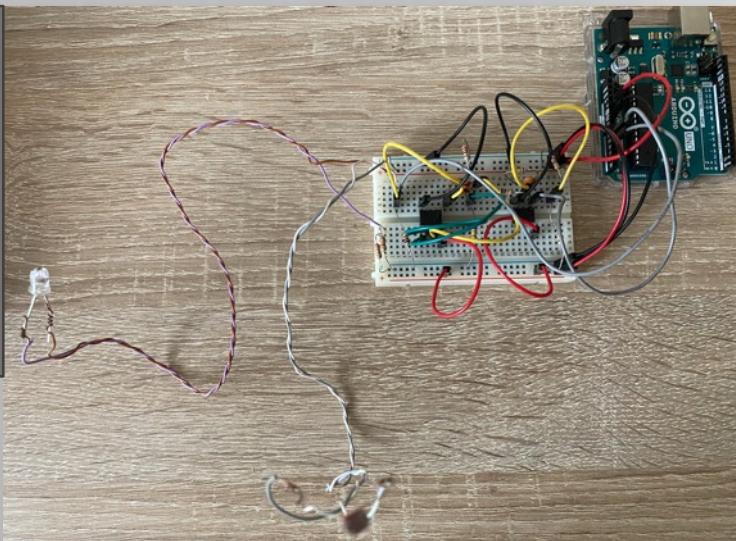


VUE GÉNÉRALE DU MONTAGE DU CIRCUIT

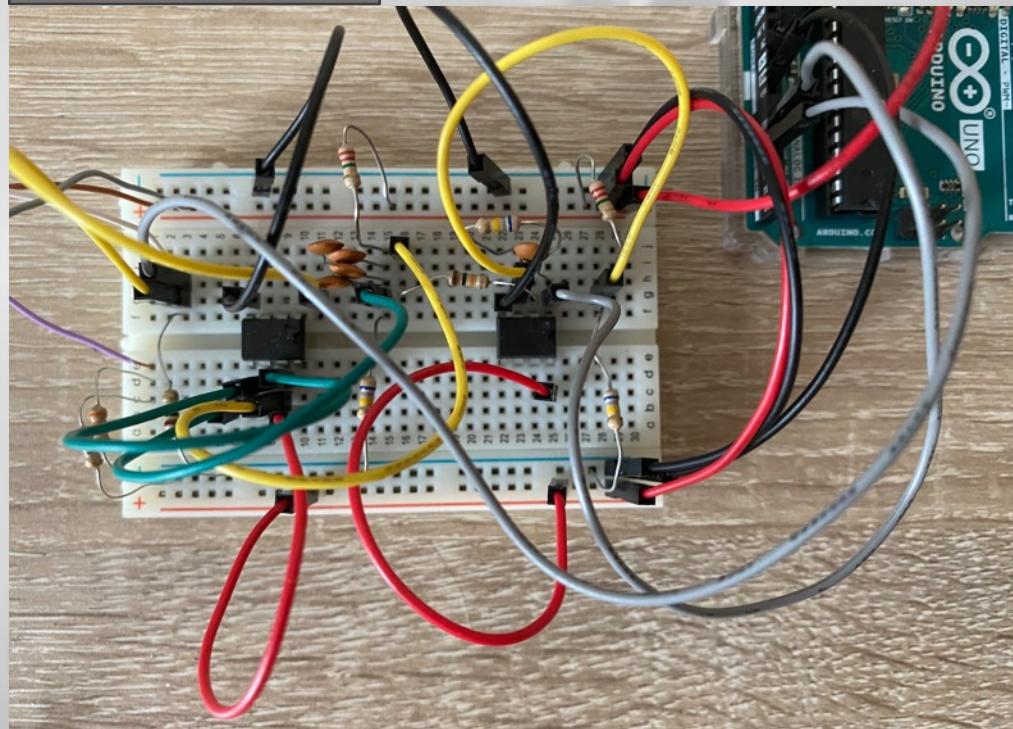
Montage complet



Circuit et carte Arduino

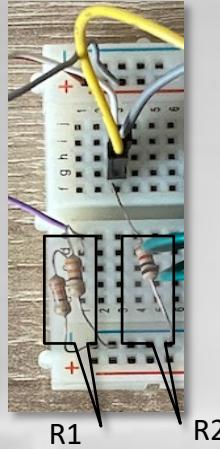
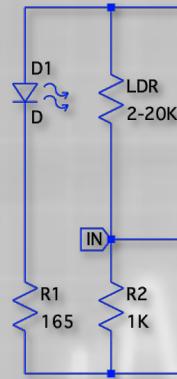
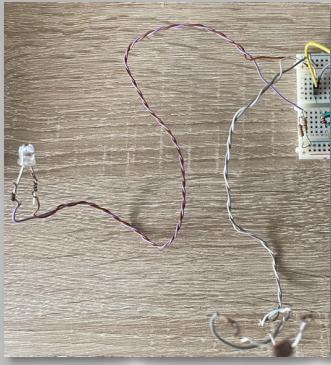


Zoom sur le circuit



EQUATIONS DES ÉTAGES DU CIRCUIT (1/2)

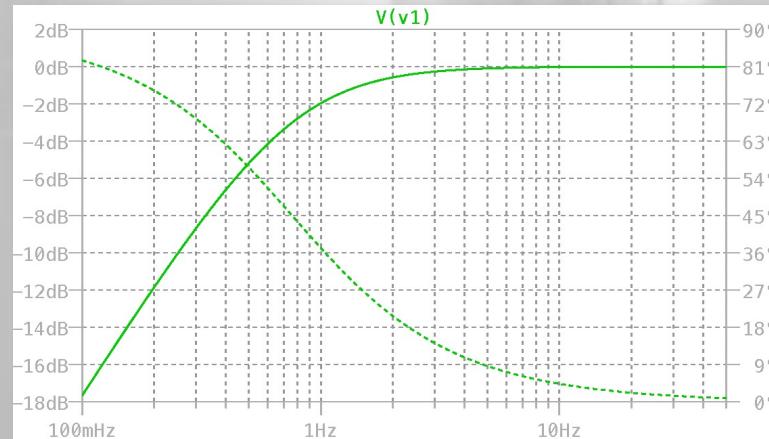
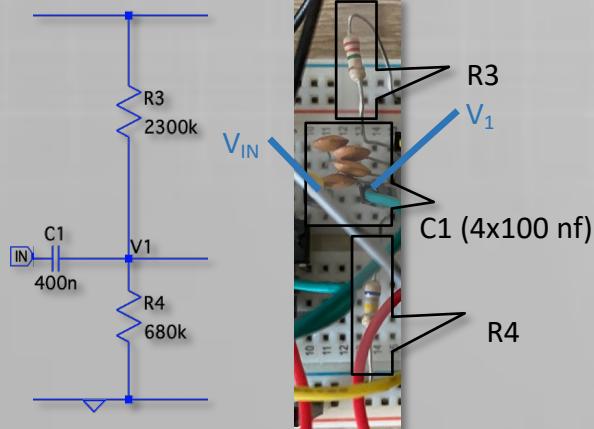
Eclairage et capture du signal



❖ $V_{IN} = E \frac{R_2}{LDR + R_2}$ ($E=5V$).

❖ (V_{IN} décroît au moment de la systole car LDR croît)

Filtre Passe-Haut et Tension Offset



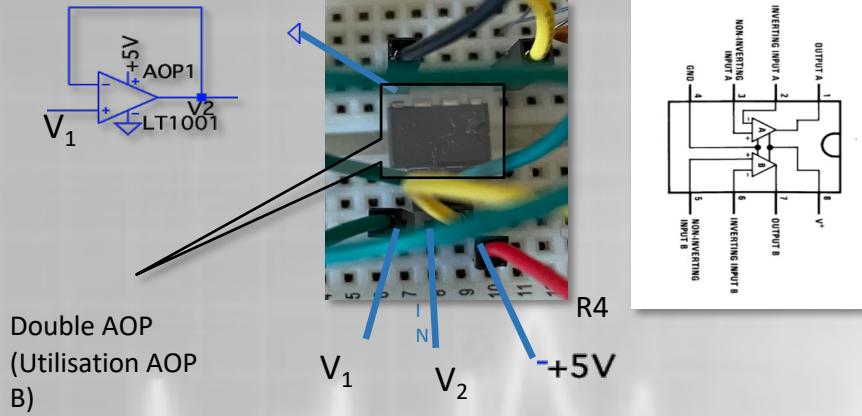
❖ $\omega_0 = \frac{R_3 + R_4}{R_3 R_4 C_1} = 4,76 s^{-1}$

❖ $f_{c0} = \frac{\omega_0}{2\pi} = 0.76 \text{ Hz}$

❖ $V_1 = V_{off} + V_{IN(\omega>0)} \frac{1}{1-j\frac{\omega_0}{\omega}}$

EQUATIONS DES ÉTAGES DU CIRCUIT (2/2)

Suiveur de tension (adaptateur d'impédance)

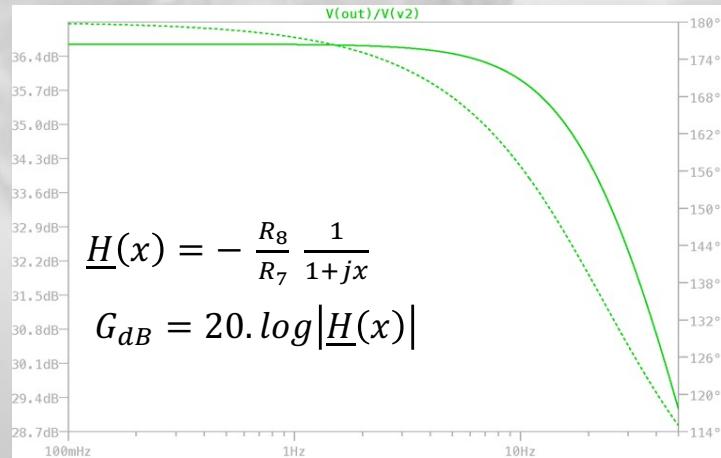
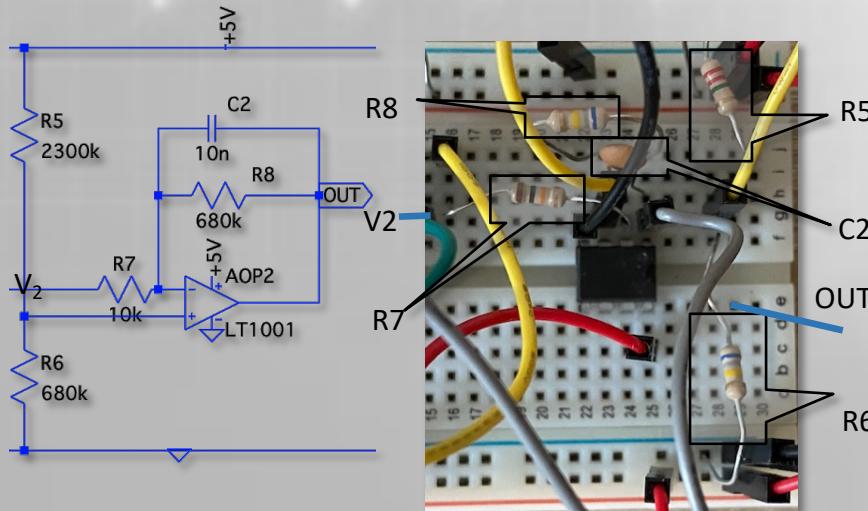


❖ Boucle de contre-réaction

$$V_2 = V_1 = V_{off} + V_{IN(\omega>0)} \frac{1}{1-j\frac{\omega_0}{\omega}}$$

❖ Impédance de sortie de l'AOP (V_2) proche de 0 Ohm
→ Signal non perturbé par les composants à l'aval.

Filtre passe-bas et amplification



$$\underline{H}(x) = - \frac{R_8}{R_7} \frac{1}{1+jx}$$

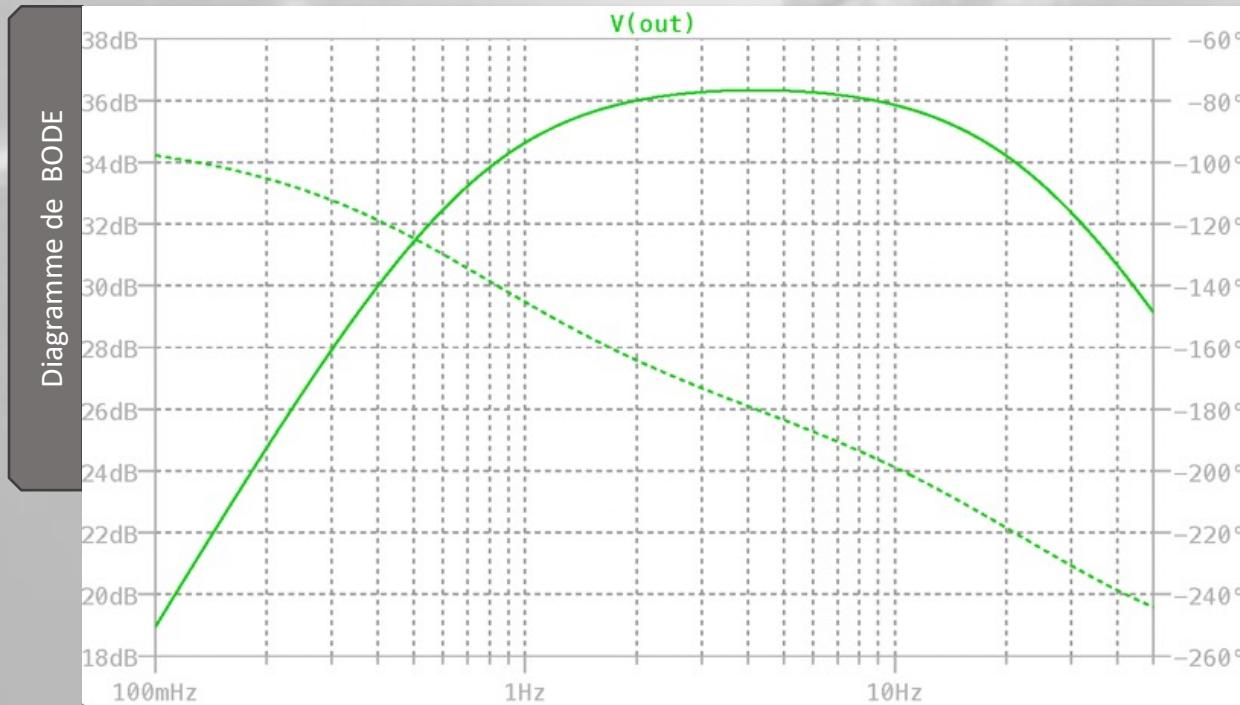
$$G_{dB} = 20 \cdot \log |\underline{H}(x)|$$

$$❖ \omega_1 = \frac{1}{R_8 C_2} \approx 147 \text{ s}^{-1} \text{ et } f_{c1} = \frac{\omega_0}{2\pi} \approx 23 \text{ Hz}$$

SYNTHÈSE GÉNÉRALE DU CIRCUIT

$$\text{Donc } V_{OUT}(\omega) = V_{off} - \frac{R_8}{R_7} \frac{1}{1+j\frac{\omega}{\omega_1}} \frac{1}{1-j\frac{\omega_0}{\omega}} V_{IN}(\omega>0)$$

Filtre Passe Bande entre ω_0 et ω_1 avec amplification/inversion du signal pulsatile de V_{IN}
et ajout d'une tension continue V_{off}





II - PROGRAMMATION DU TRANSFERT, DE L'ENREGISTREMENT ET DE L'AFFICHAGE DES SIGNAUX OBTENUS SUR MICRO-ORDINATEUR ESSAI DU MONTAGE

PRINCIPES ET CONCEPTION DES PROGRAMMES



❖ Programmer la carte Arduino :

- pour échantillonner les signaux Vin et Vout [sur 8 bits – 1024 valeurs], à une fréquence de 200Hz,
- Puis transmettre les données par couple via un câble USB à l'ordinateur en vue de leur affichage ou traitement

❖ Programmation dans un logiciel fourni

❖ Langage C

❖ Coder `setup()` et `loop()`

❖ Programmer l'ordinateur pour :

- Lire les couples de données sur le port USB
- Animer le tracé des deux courbes
- Stocker sur disque l'ensemble des données capturées dans des fichiers texte CSV

❖ Programmation en Python

❖ Librairies pySerial et matplotlib

❖ Animation avec `funcAnimation`

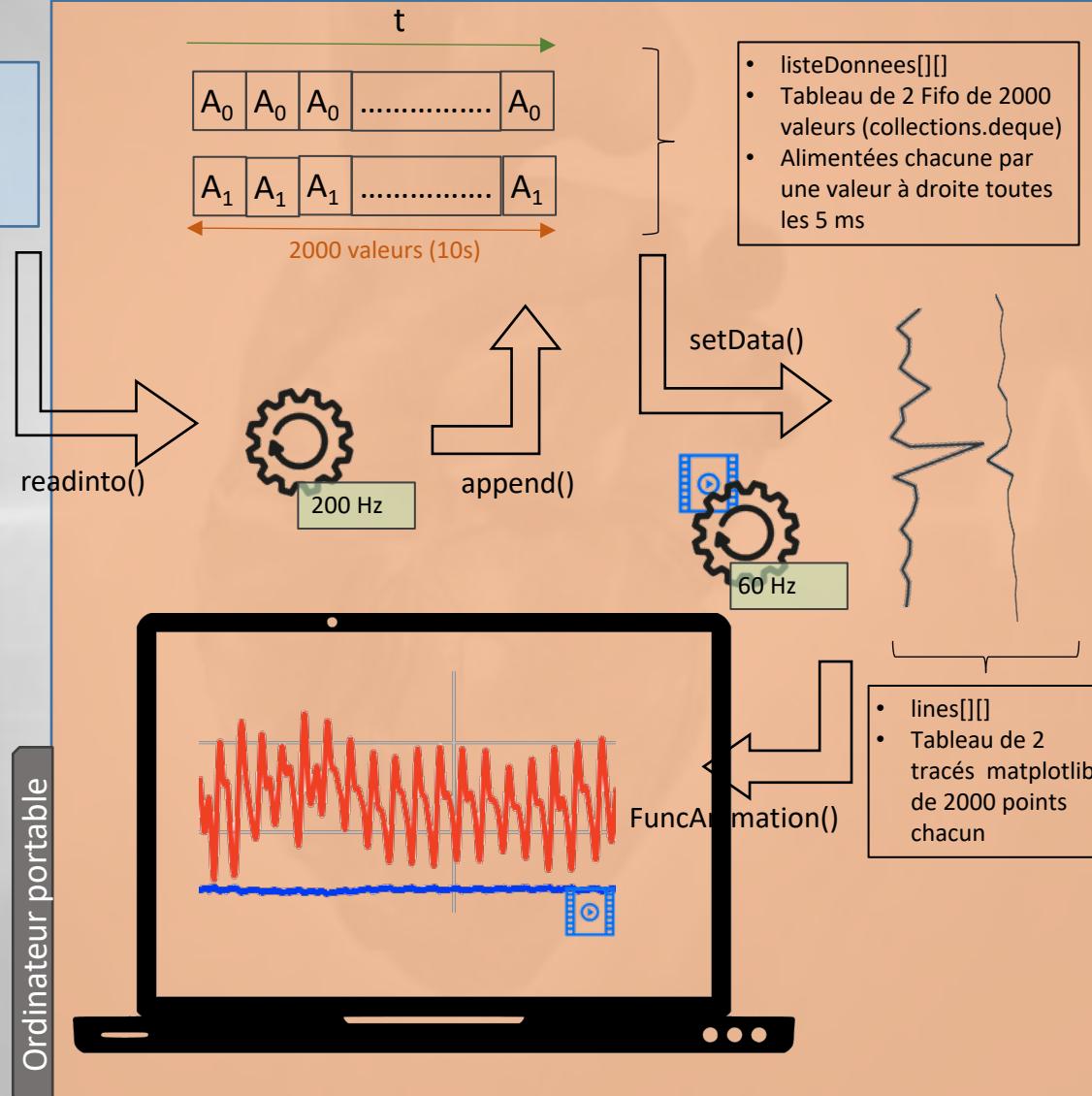
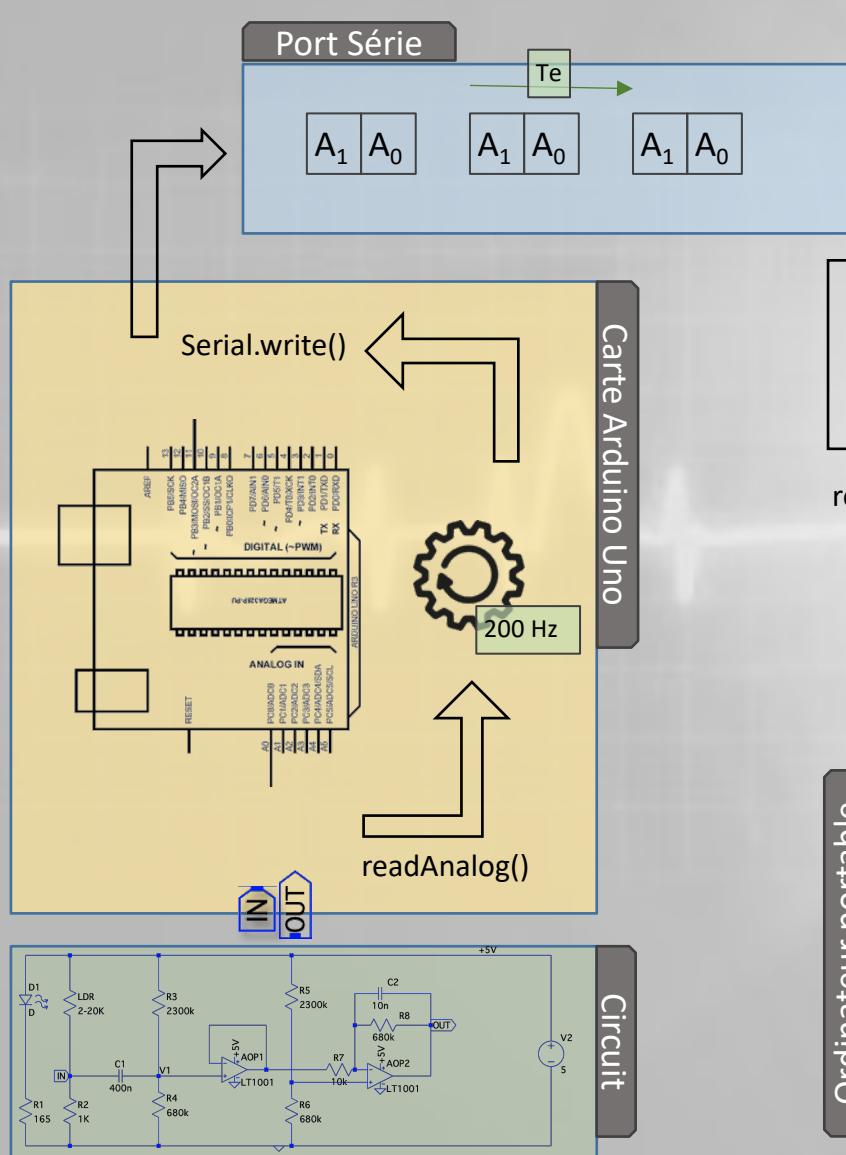
PROGRAMMATION DE L'ORDINATEUR – PROBLÉMATIQUE RENCONTRÉE

- 
- ❖ La fonction `matplotlib animation.FuncAnimation` est synchronisée avec la fréquence de rafraîchissement de l'écran de l'ordinateur (inutile de s'exécuter plus vite), soit 50 ou 60 Hz.

=> Elle ne doit donc pas se situer au sein de la boucle de lecture des données sur le port USB qui a besoin d'une rapidité de 200Hz, sous peine de l'apparition immédiate d'un retard d'affichage

- ❖ Ceci a nécessité la programmation d'un fil d'exécution (appelé `thread`) indépendant pour la lecture des données USB et leur positionnement dans un tableau lu par la fonction d'animation.

SCHÉMA DE SYNTHÈSE (MULTI-PROCESSUS)



ESSAI DU MONTAGE ET DES PROGRAMMES



- ❖ Plaquer LED et Photo-Résistance sur le bout d'un doigt, a 1cm l'une de l'autre
- ❖ Ne surtout pas bouger (le moindre mouvement introduit de la lumière parasite sur le signal)
- ❖ Observer à l'écran le tracé animé des 10 dernières secondes des deux signaux
- ❖ Récupérer les deux fichiers CSV des deux signaux pour traitement ultérieur

Création Thread lecture données

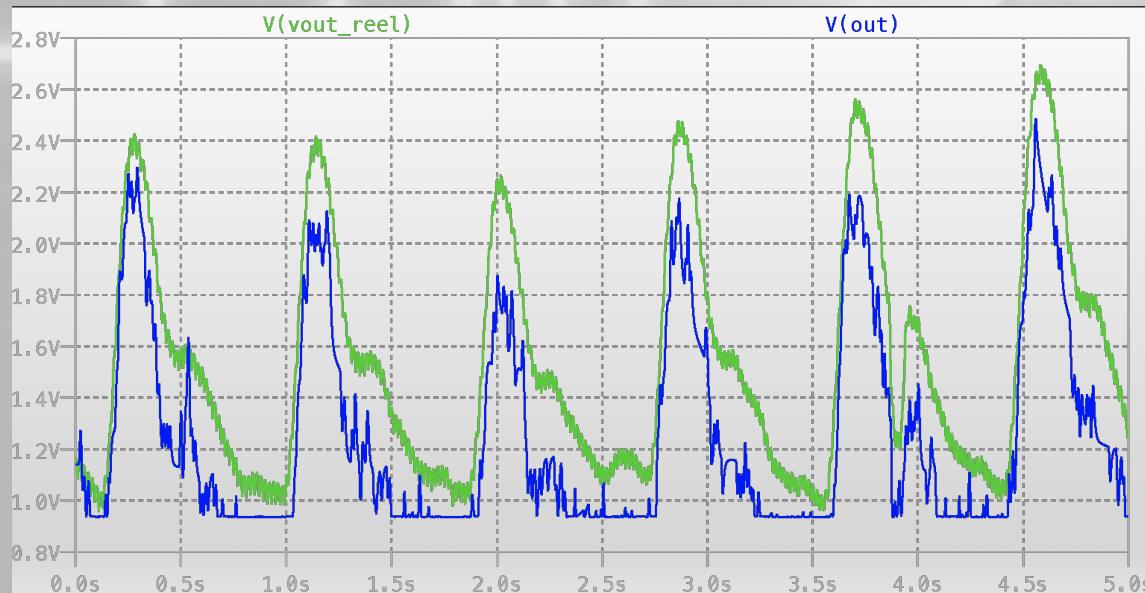
```
if self.processusLectureDonnees == None:  
    self.processusLectureDonnees = Thread(target=self.creerProcessusLectureDonnees())  
    self.processusLectureDonnees.start()  
  
# Si le processus n'a pas été lancé
```

Si le processus n'a pas été lancé

On le lance

COMPARAISON SIMULATION / CIRCUIT RÉEL

- ❖ Injecter le signal brut en entrée dans LTSPICE, et exécuter le circuit (pendant 5 secondes)
- ❖ Injecter aussi (sur un autre générateur virtuel totalement indépendant) le signal traité par le circuit.
- ❖ Superposer alors pour les comparer les deux signaux : $V_{\text{out}}^{\text{Simulé}}$ et $V_{\text{out}}^{\text{réel}}$.



- ❖ Signaux très proches
- ❖ Signal analogique plus « continu »
- ❖ Analogique : Haute fréquence parasite apparaissant probablement après le filtre passe haut (fil allant du circuit vers la carte)

III - CONCEPTION ET PROGRAMMATION D'UN TRAITEMENT NUMÉRIQUE DES SIGNAUX EN VUE D'OBTENIR UN CARDIOGRAMME ET UN GRAPHE DU RYTHME RESPIRATOIRE

PRÉAMBULES – QUELQUES DÉFINITIONS

On considère $x(t)$ un signal continu, échantillonné à la fréquence f_e

- ❖ On appelle *signal discret associé* la suite $(x_n)_{n \in \mathbb{Z}}$ telle que

$$x_n = x(n \cdot T_e) \text{ (avec } T_e = \frac{1}{f_e})$$

- ❖ On appelle *impulsion* un signal discret $(x_n)_{n \in \mathbb{N}}$ défini par :

$$\begin{cases} x_0 = 1 \\ x_n = 0 \text{ pour } n > 0 \end{cases}$$

- ❖ Le *filtrage numérique linéaire causal* consiste à obtenir un nouveau signal numérique y_n en utilisant une relation de la forme suivante :

$$y_n = \sum_{k=0}^{n-1} h_k x_{n-k} \text{ avec } (h_k)_{k \in \mathbb{N}} \text{ une suite réelle}$$

- ❖ On dit qu'un filtre est *stable* si $\lim_{n \rightarrow +\infty} h_n = 0$. Ainsi si $(h_k)_{k \in \mathbb{N}}$ est une suite réelle presque nulle, le filtre est stable et exploitable physiquement.

FILTRES NUMÉRIQUES PASSE-BANDE

- ❖ Soit un signal discret $(x_n)_{n \in \mathbb{N}}$, échantillonné à la fréquence f_e
- ❖ On démontre par analyse/synthèse (Cf. Annexes), que les coefficients de la suite $(g_n)_{n \in \mathbb{Z}}$, définis par:

$$g_k = \frac{1}{k\pi} \sin\left(2k\pi \frac{f_c}{f_e}\right), -P \leq k \leq P, P \in \mathbb{N}$$

$g_k = 0$ sinon

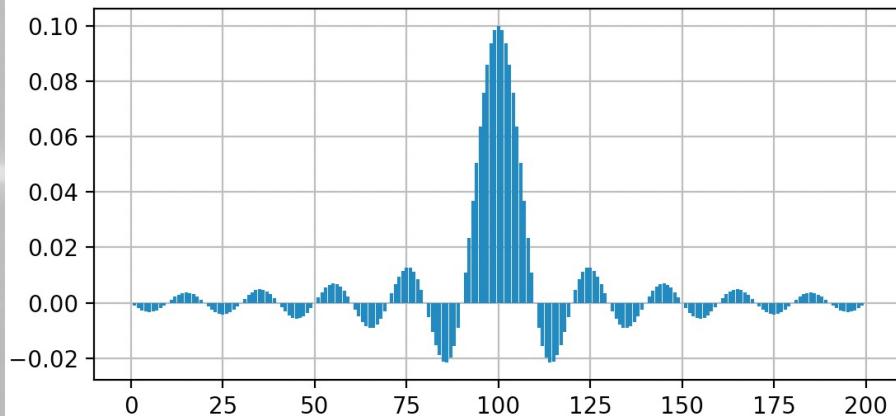
Sont les $2P+1$ coefficients d'un filtre numérique linéaire causal passe-bas de fréquence de coupure f_c , et de gain 1.

- ❖ Le filtre est stable car $(g_n)_{n \in \mathbb{Z}}$ est presque nulle.
- ❖ P est appelé l'ordre du filtre.
- ❖ Le filtre a un retard structurel de PT_e secondes par rapport à l'entrée.
- ❖ On déduit de même les coefficients pour les filtres passe-haut (signe négatif de g_k) et passe bande (somme des g_k)

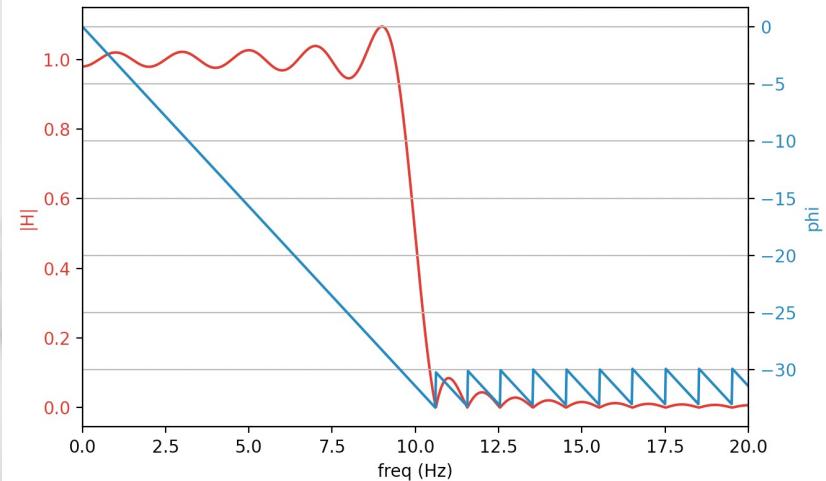
EXEMPLE D'UN FILTRE PASSE BAS CODÉ EN PYTHON

- ❖ En programmant en python un filtre passe bas de fréquence 10 Hz, avec $P=100$, on obtient :

Réponse impulsionnelle



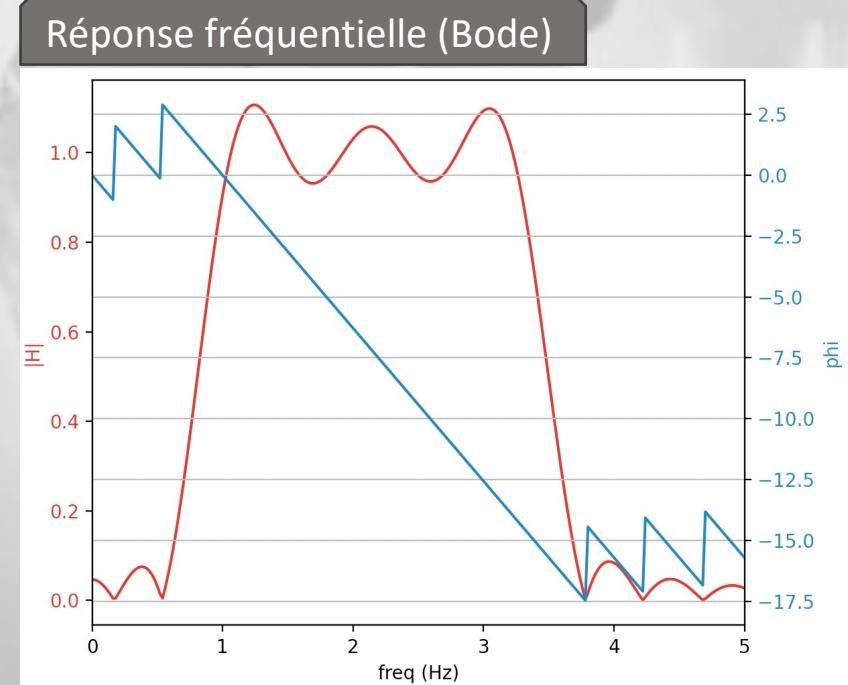
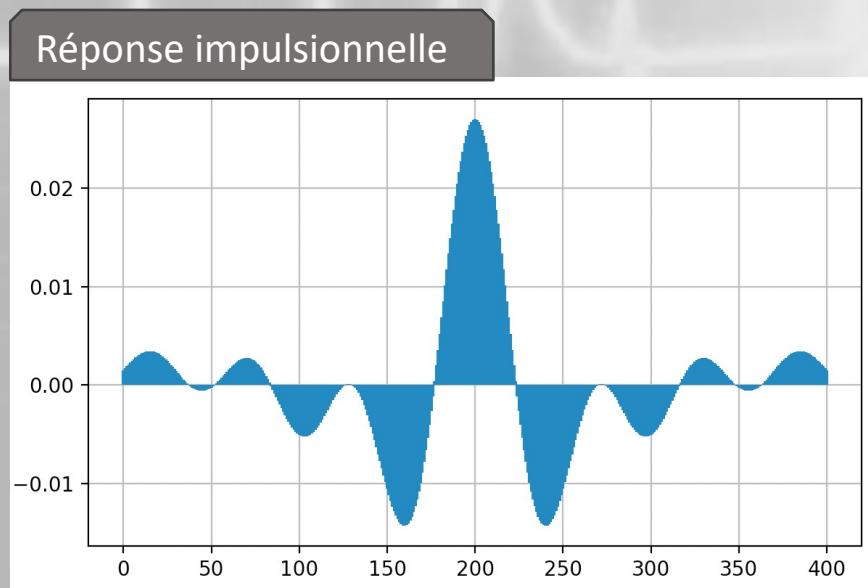
Réponse fréquentielle (Bode)



- ❖ Nous observons bien un gain de 1 (oscillant) dans la BP, ainsi qu'une phase linéaire dans la BD.
- ❖ Nous allons programmer ces filtres pour nos signaux PPG

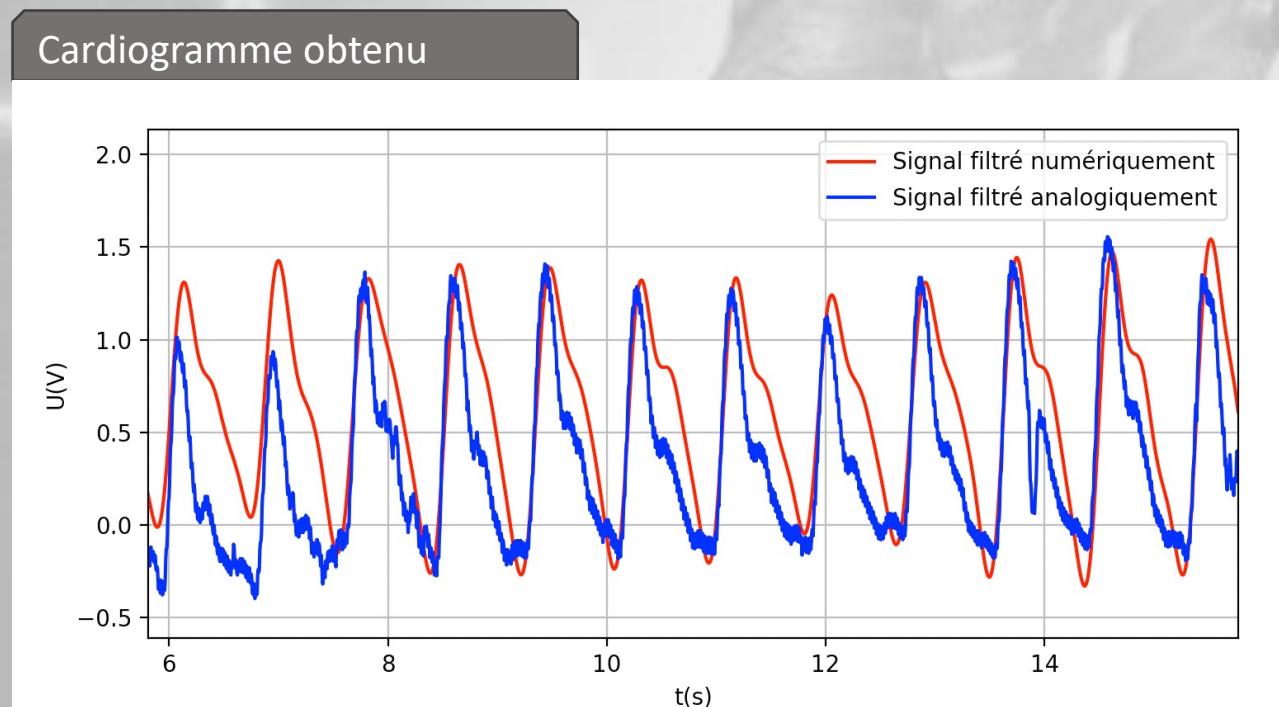
FILTRE PASSE-BANDE POUR OBTENIR LE CARDIOGRAMME (1/2)

- ❖ Notre signal brut a été échantillonné à 200Hz.
- ❖ Programmer un filtre Passe Bande entre 0,75Hz et 3Hz, avec P=200



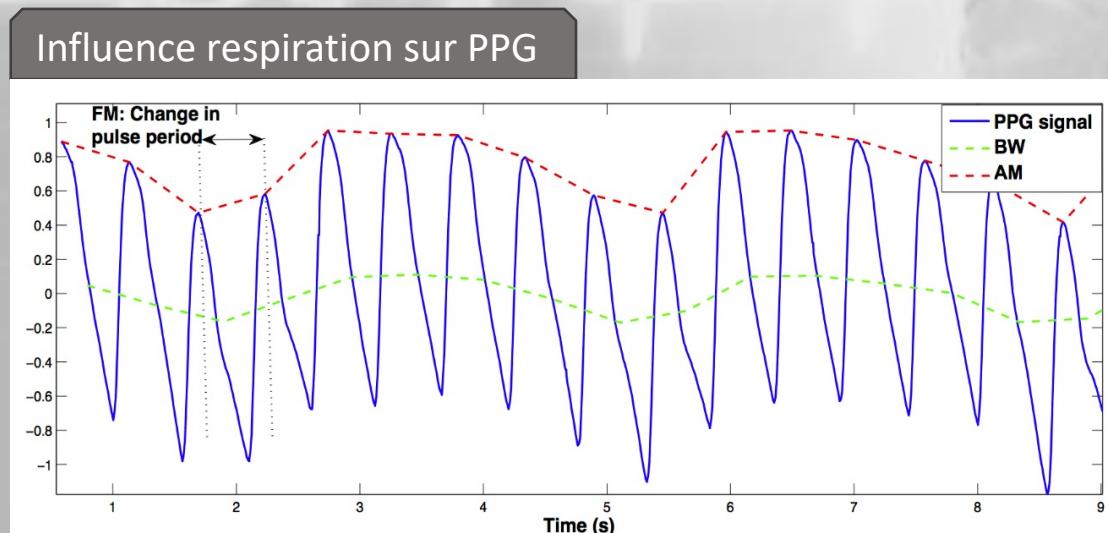
FILTRE PASSE-BANDE POUR OBTENIR LE CARDIOGRAMME (1/2)

- ❖ Notre signal a un retard de $PT_e = 1s$ sur l'entrée
- ❖ Décaler d'une seconde et l'afficher sur le même graphe que le signal traité analogiquement (avec ajustement gain et offset)



RECHERCHE DU RYTHME RESPIRATOIRE (1/4)

- ❖ La littérature indique que signal cardiaque PPG est influencé par la fréquence respiratoire de 3 manières
 - Baseline Wander (fluctuation de la ligne de base)
 - Amplitude Modulation (modulation de l'amplitude du signal)
 - Frequency Modulation (modulation de la fréquence cardiaque) = arythmie sinusale: le rythme croît à l'inspiration et décroît à l'expiration



D'après Sala Cherif - Effective signal processing methods for robust respiratory rate estimation from photoplethysmography

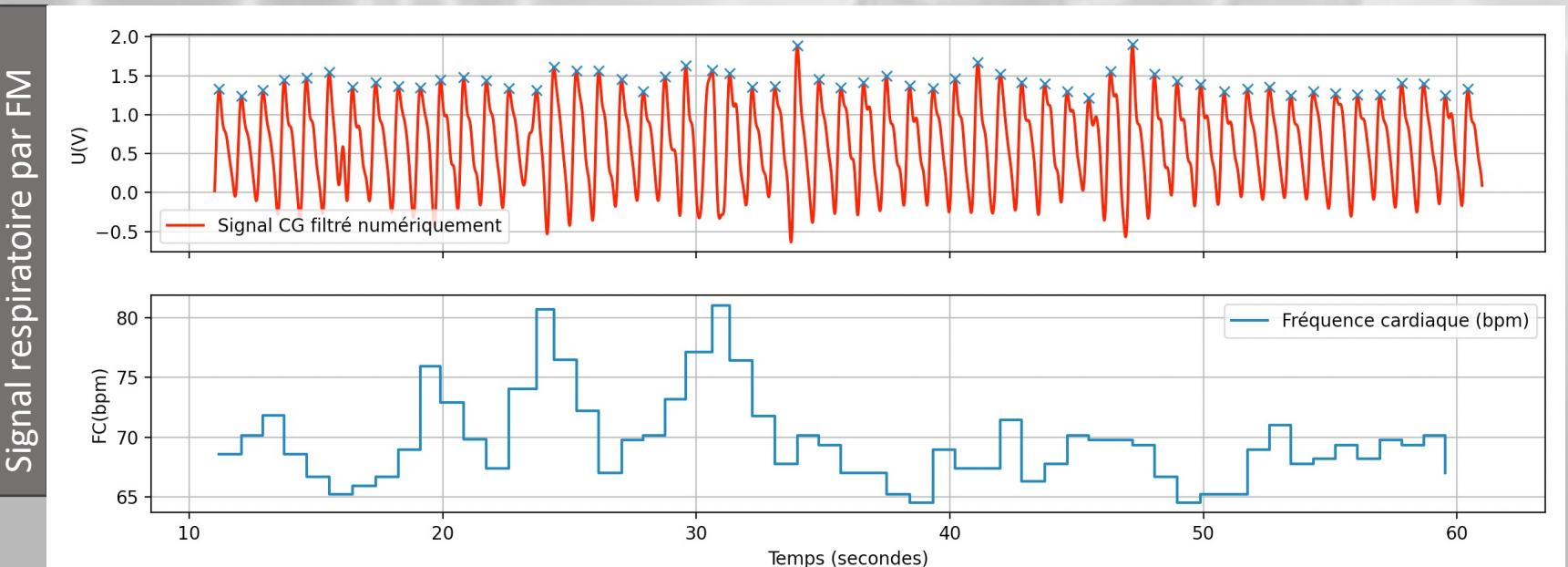
Essayons de détecter selon :

- Frequency Modulation
- Baseline Wander

RECHERCHE DU RYTHME RESPIRATOIRE PAR ARYTHMIE SINUSALE(2/4)

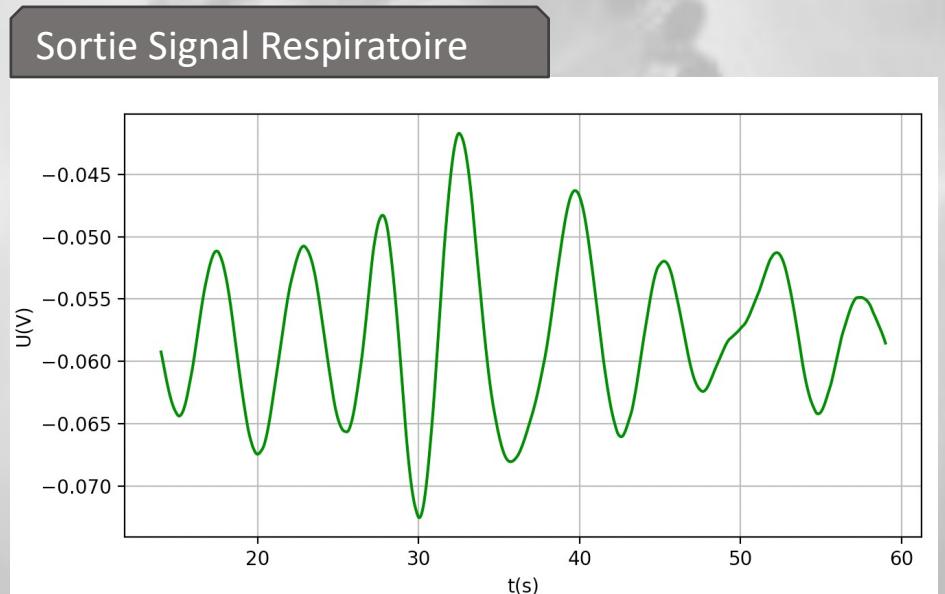
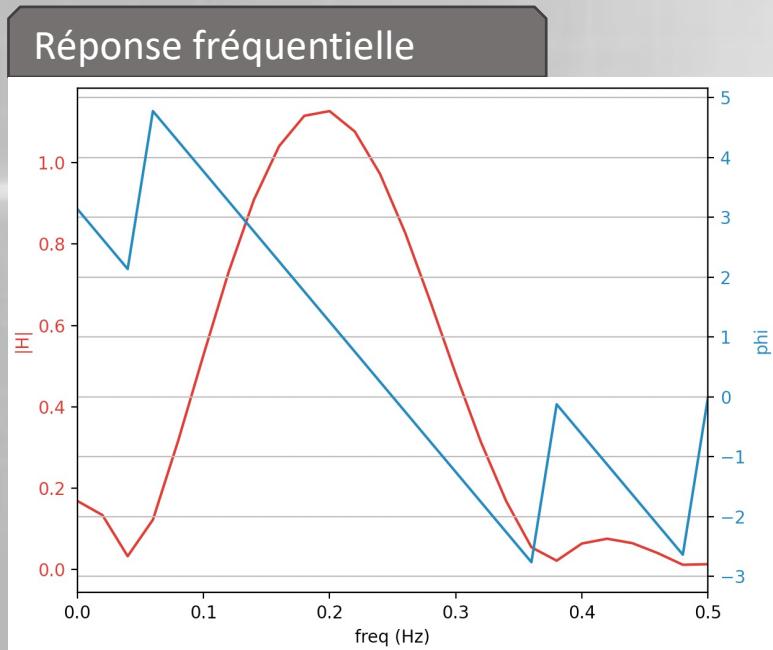
- ❖ On programme en python une détection des pics (`find_peaks`) sur le PPG
- ❖ On en déduit le rythme cardiaque, dans lequel on corrige les erreurs manifestes (point aberrant remplacé par une interpolation linéaire entre point précédent et suivant)
- ❖ On trace alors le diagramme des fréquences cardiaques qui montre une période de l'ordre de 5s

```
peaks,_ = (find_peaks(ecg[10*fe:60*fe], distance=0.6*fe))
```



RECHERCHE DU RYTHME RESPIRATOIRE PAR FILTRE NUMÉRIQUE(3/4)

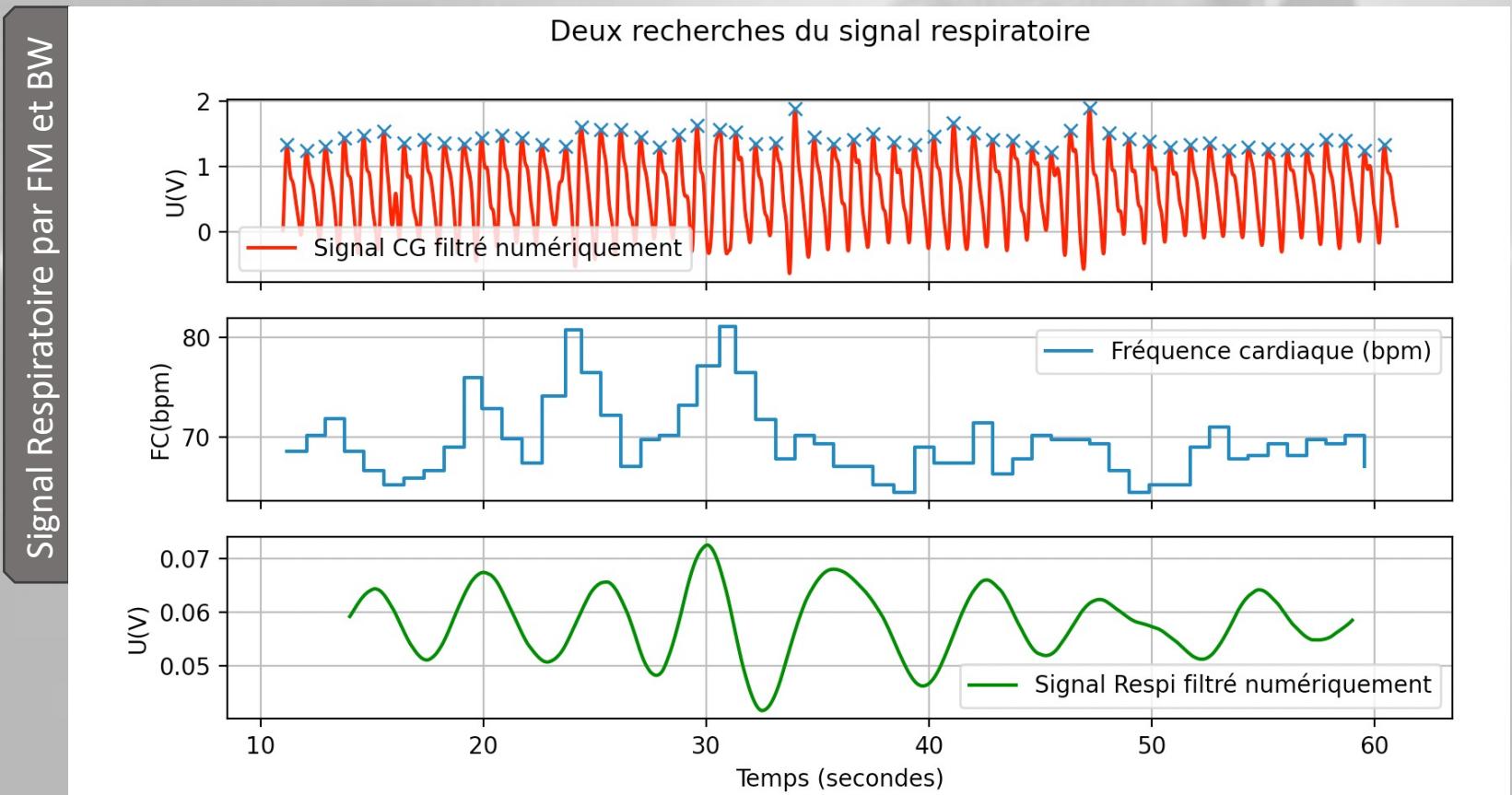
- ❖ On utilise la propriété BW (Base Wander)
- ❖ Cela signifie qu'un signal sinusoïdal de fréquence $0.1 \leq f \leq 0.3 \text{ Hz}$ (de faible amplitude) est présent dans le PPG
- ❖ On construit un filtre très sélectif, dans cette BP, avec $P=800$ (4s de retard)



- ❖ On observe bien un signal (respiratoire) de période 5s environ

RECHERCHE DU RYTHME RESPIRATOIRE PAR FILTRE NUMÉRIQUE(4/4)

❖ Synthèse des traitements de la recherche du signal respiratoire :



CONCLUSION

- ❖ Un matériel très accessible permet l'obtention d'un signal PPG permettant d'isoler un cardiogramme et la fréquence respiratoire,
- ❖ Pour être complet, il faudrait rechercher un signal respiratoire à l'effort et vérifier que sa fréquence est plus haute,
- ❖ Le filtrage analogique est facile et fonctionnel en temps réel,
- ❖ Le filtrage numérique permet la réalisation de filtres très sélectifs (au prix d'un certain retard, potentiellement problématique en temps réel),
- ❖ La PPG par webcam devient disponible, les montres connectées les plus évoluées seraient bientôt utilisées pour détecter la COVID -19

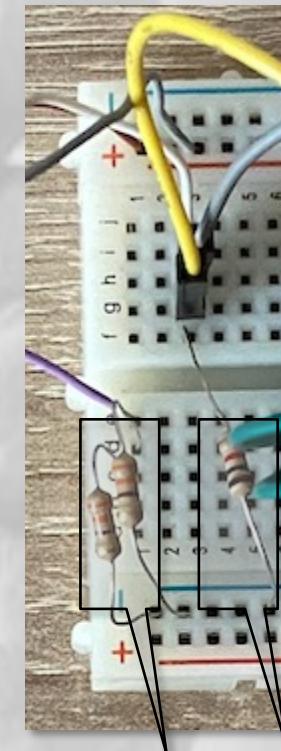
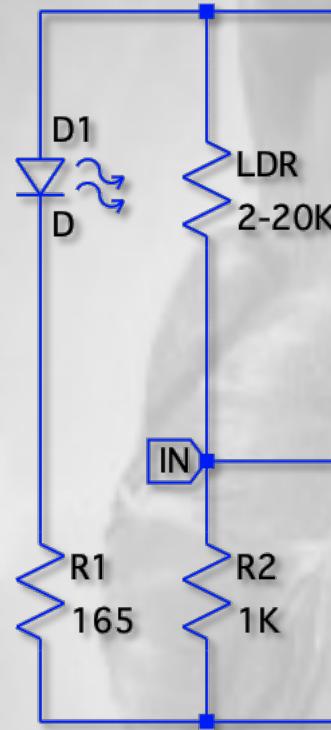
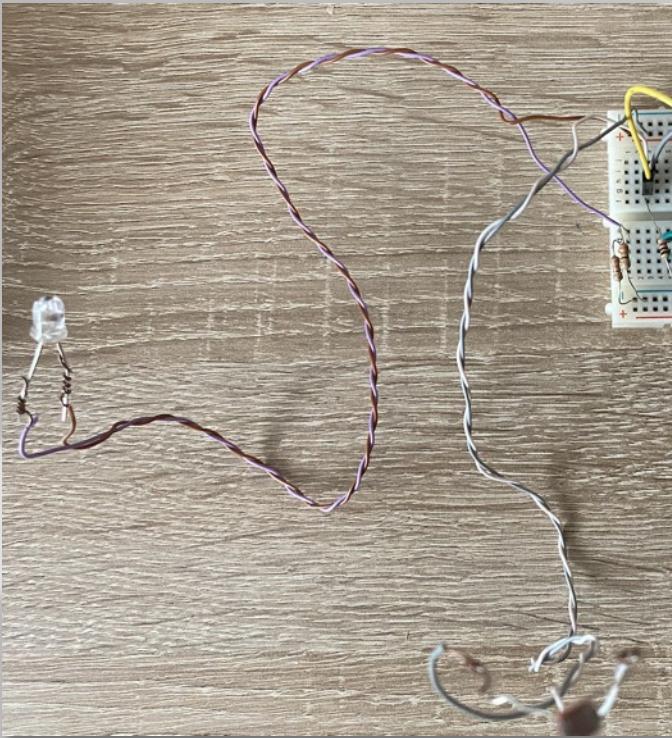
(<https://www.futura-sciences.com/tech/actualites/montre-connectee-montres-connectees-utiles-detecter-covid-19-avant-symptomes-80459/>)

ANNEXES

ANNEXE 1

Démonstrations des équations du circuit

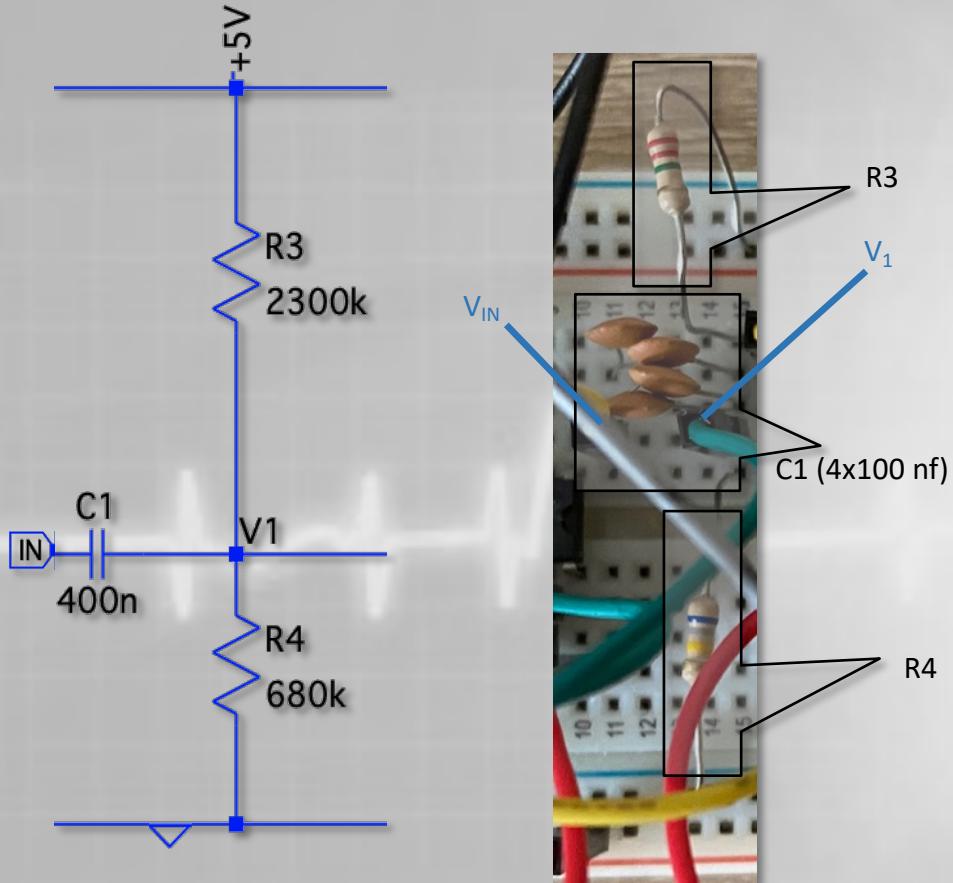
PARTIE ÉCLAIRAGE ET GÉNÉRATION DU SIGNAL



R1 R2

- ❖ R_1 permet une intensité de 30,3 mA dans D_1 , conforme aux spécifications
- ❖ R_2 est une résistance de sécurité
- ❖ $V_{IN} = E \frac{R_2}{LDR + R_2}$ ($E=5V$). (V_{IN} décroît au moment de la systole car LDR croît)

PARTIE FILTRE PASSE-HAUT ET OFFSET



$$\diamond V_{1(\omega)} = E(\omega) \frac{R_4}{R_3+R_4} \frac{1}{1+j\frac{\omega}{\omega_0}} + V_{IN(\omega)} \frac{1}{1-j\frac{\omega_0}{\omega}}$$

$$\diamond \omega_0 = \frac{R_3+R_4}{R_3 R_4 C_1} = 4,76 s^{-1}$$

$$\diamond f_{c0} = \frac{\omega_0}{2\pi} = 0.76 \text{ Hz}$$

❖ Partie continue : $\omega = 0$

$$V_{1,\omega=0} \approx E_0 \frac{R_4}{R_3+R_4} = V_{off} = 1.14 \text{ V}$$

=> Tension d'offset

❖ Partie pulsatile : $\omega > 0$

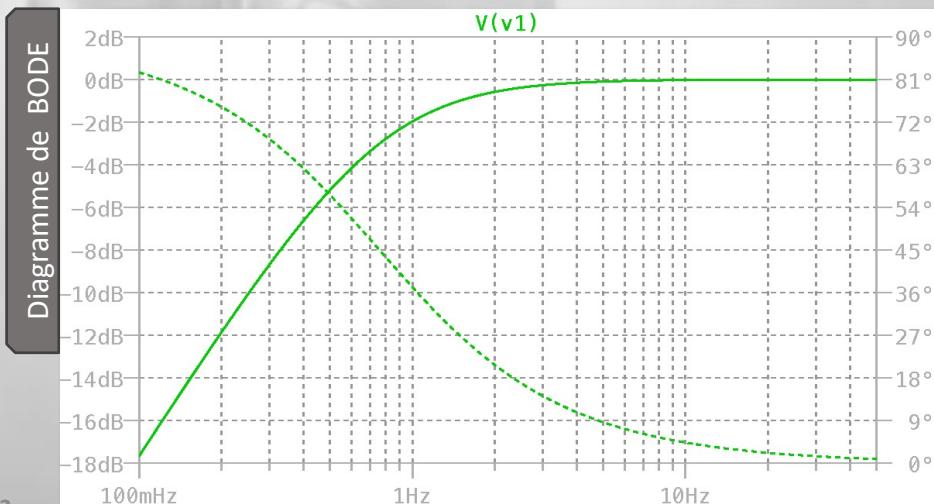
$$V_{1,\omega>0} = V_{IN(\omega>0)} \frac{1}{1 - j \frac{\omega_0}{\omega}}$$

⇒ Filtre passe haut de la partie pulsatile du signal PPG

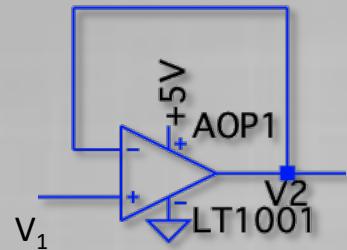
$$\Rightarrow \text{Donc } V_1 = V_{off} + V_{IN(\omega>0)} \frac{1}{1 - j \frac{\omega_0}{\omega}}$$

❖ Pour $\omega \gg \omega_0$

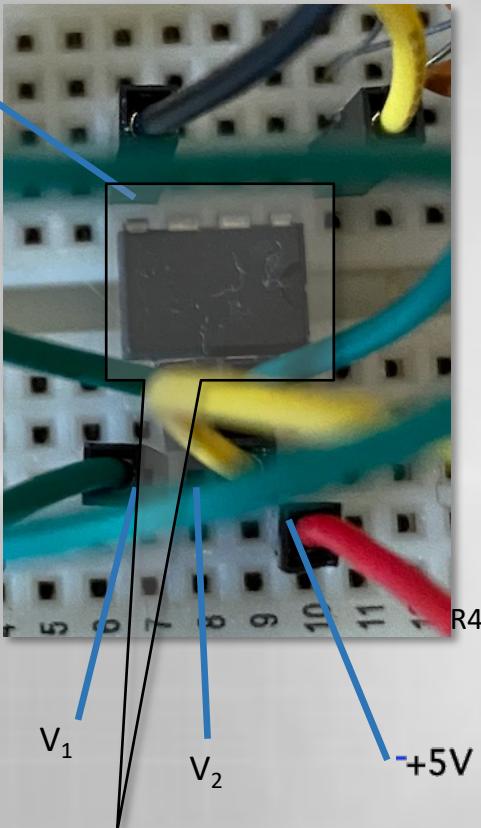
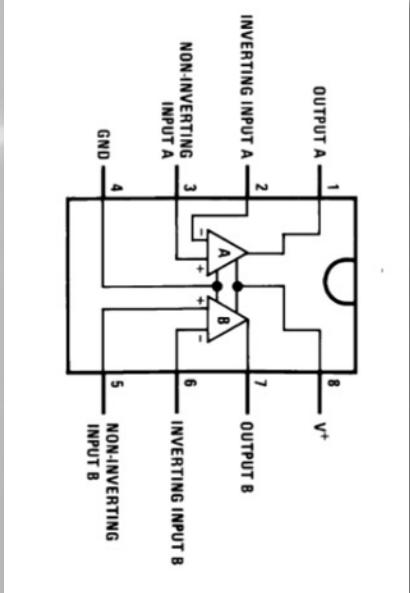
$$V_1 = V_{off} + V_{IN(\omega, \omega \gg \omega_0)}$$



PARTIE SUIVEUR DE TENSION (AD. IMPÉDANCE)



Détail puce AOP



Double AOP
Utilisation du B

❖ Boucle de contre-réaction

- Fonctionnement en mode linéaire
- $V_+ = V_-$, $i_+ = i_- = 0A$

❖ $V_2 = V_1$

$$\diamond V_2 = V_1 = V_{off} + V_{IN(\omega>0)} \frac{1}{1-j\frac{\omega_0}{\omega}}$$

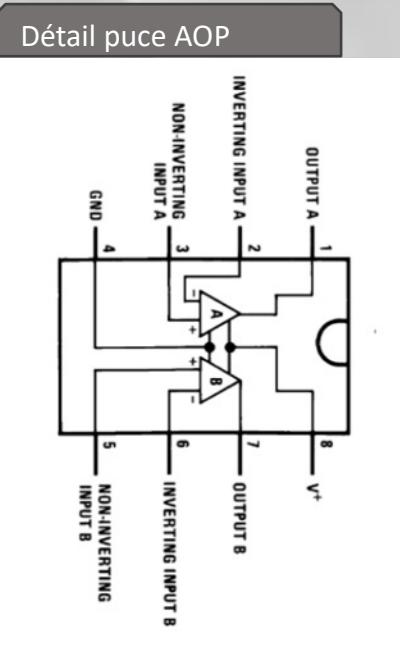
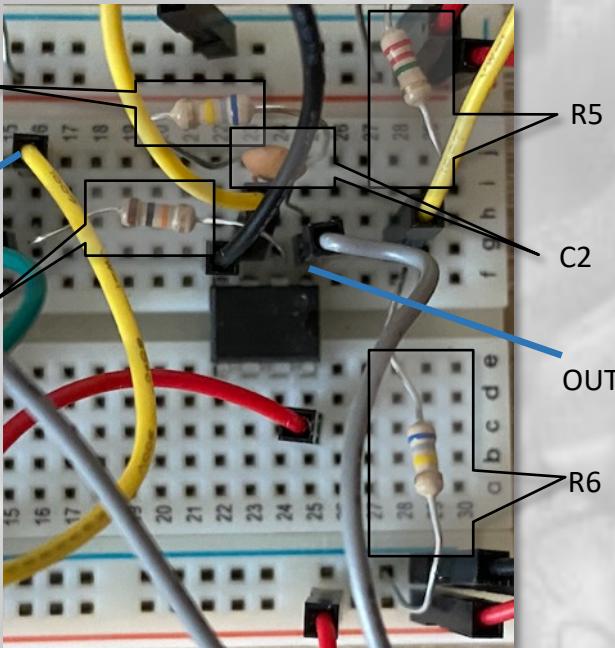
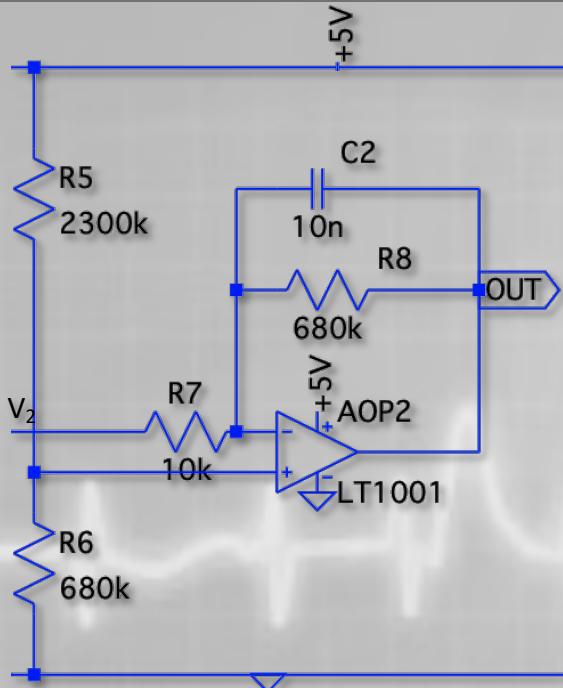
❖ Pour $\omega \gg \omega_0$

$$V_2 = V_1 = V_{off} + V_{IN(\omega,\omega \gg \omega_0)}$$

❖ Impédance de sortie de l'AOP (V_2) proche de 0 Ohm

- Signal non perturbé par les composants à l'aval.

PARTIE FILTRE PASSE-BAS ET AMPLIFICATION (1/2)



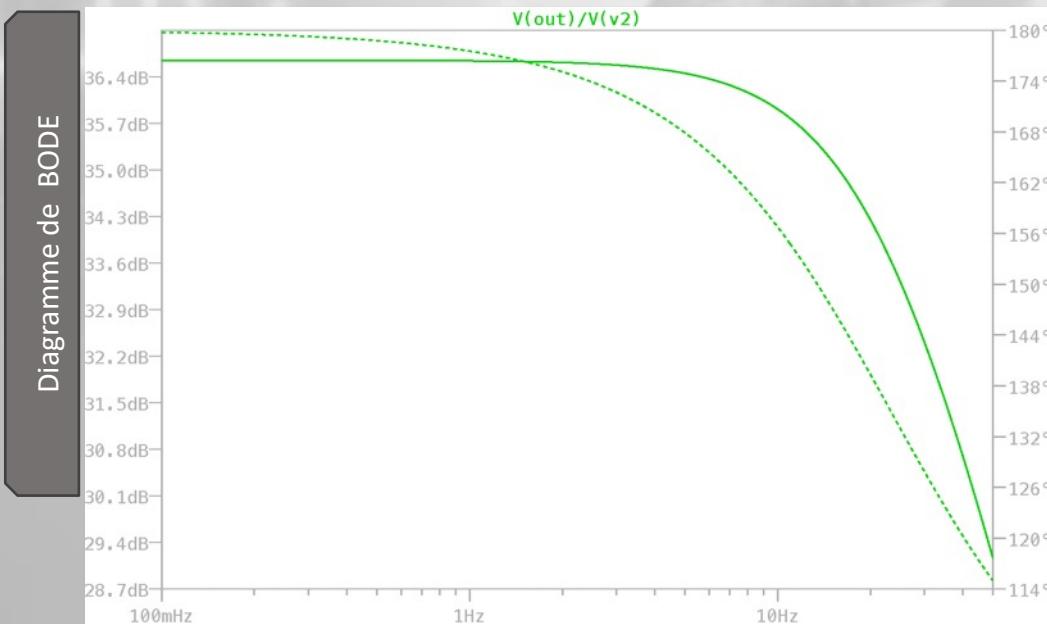
$$\diamond V_{OUT}(\omega) = E(\omega) \frac{R_6}{R_5+R_6} - \frac{R_8}{R_7} \frac{1}{1+j\frac{\omega}{\omega_1}} (V_{2(\omega)} - E(\omega) \frac{R_6}{R_5+R_6})$$

$$\diamond \omega_1 = \frac{1}{R_8 C_2} \approx 147 \text{ s}^{-1} \text{ et } f_{c1} = \frac{\omega_0}{2\pi} \approx 23 \text{ Hz}$$

$$\diamond \rightarrow V_{OUT}(\omega) = V_{off} - \frac{R_8}{R_7} \frac{1}{1+j\frac{\omega}{\omega_1}} (V_{2(\omega)} - V_{off}) \text{ car } E(\omega) \frac{R_6}{R_5+R_6} = V_{off}$$

PARTIE FILTRE PASSE-BAS ET AMPLIFICATION (2/2)

- ❖ Pour $\omega > 0$, $\rightarrow V_{OUT(\omega>0)} = - \frac{R_8}{R_7} \frac{1}{1+j\frac{\omega}{\omega_1}} (V_2(\omega>0))$
- ❖ Filtre passe bas actif – amplificateur – inverseur (systole vers le haut)
- ❖ $H(x) = - \frac{R_8}{R_7} \frac{1}{1+jx}$
- ❖ Gain : $G_{dB} = 20 \cdot \log |H(x)| = 20 \cdot \log \left(\frac{R_8}{R_7} \frac{1}{\sqrt{1+x^2}} \right) \approx 36,6 - 10 \cdot \log(1 + x^2)$

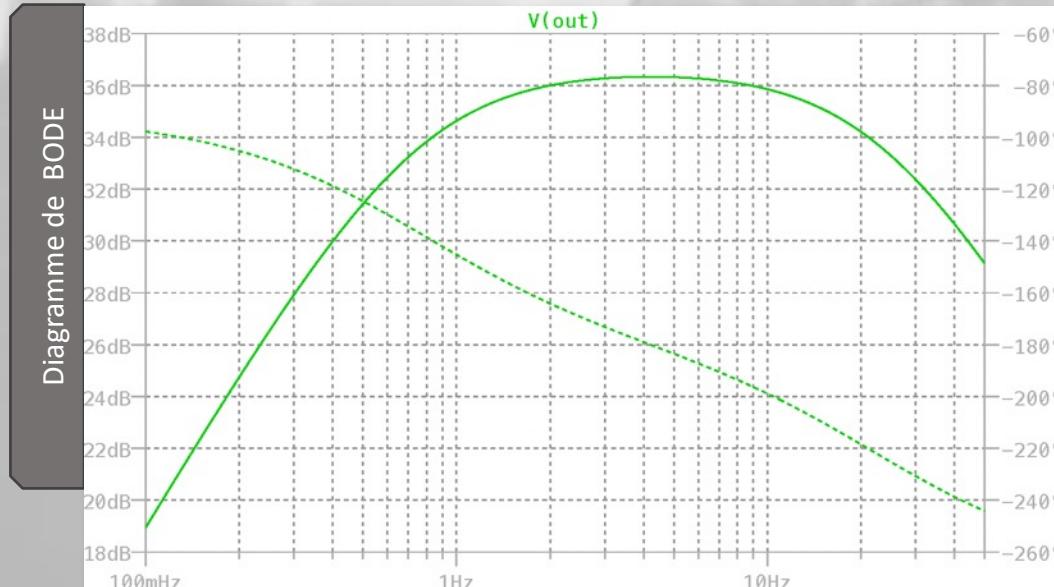


SYNTHÈSE GÉNÉRALE DU CIRCUIT

- ❖ $V_{OUT}(\omega) = V_{off} - \frac{R_8}{R_7} \frac{1}{1+j\frac{\omega}{\omega_1}} (V_{2(\omega)} - V_{off})$
- ❖ Mais $V_{2(\omega)} = V_{1(\omega)} = V_{off} + V_{IN(\omega>0)} \frac{1}{1-j\frac{\omega_0}{\omega}}$

$$\text{Donc } V_{OUT}(\omega) = V_{off} - \frac{R_8}{R_7} \frac{1}{1+j\frac{\omega}{\omega_1}} \frac{1}{1-j\frac{\omega_0}{\omega}} V_{IN(\omega>0)}$$

Filtre Passe Bande entre ω_0 et ω_1 avec amplification/inversion du signal pulsatile de V_{IN}
et ajout d'une tension continue V_{off}





ANNEXE 2

Principes de programmation utilisés

LANGAGES ET LIBRAIRIES UTILISÉES

- ❖ Environnement de programmation fourni avec la carte
- ❖ Programme écrit en langage C, avec deux fonctions obligatoires :
 - Setup() : Exécuté à l'initialisation de la carte
 - Loop() : programme principal exécuté en boucle par le microprocesseur de la carte, à 8MHz.
- ❖ Fonctions disponibles : <https://www.arduino.cc/reference/fr/>
- ❖ Contenu du loop de notre programme
 - Attendre un délai de $T_e = 5\text{ms}$ ($f_e=200\text{ Hz}$) depuis l'entrée précédente dans loop()
 - Relever les valeurs analogiques sur les pin A0 et A1 (2 entiers de 2 octets)
 - Transformer ces valeurs en 2 tableaux de 2 octets
 - Envoyer les 4 octets concaténés sur le port série (valeurs envoyées forcément en couple)

```
void loop() {
    attendreTop(te);
    int val0 = analogRead(0);
    int val1 = analogRead(1);
    ecrireDonnees(&val0, &val1);
}
```

```
void ecrireDonnees(int* val0, int* val1)
{
    // Serial.write prend un tableau d'octets en argument, ainsi que la longueur à écrire
    // On convertit donc les entiers (2 octets) en tableaux de bytes, puis on concatène avant envoi
    // Il est ainsi impossible d'avoir une valeur de A0 et une valeur de A1 envoyées séparément
    byte* byteVal0 = (byte*)(val0);
    byte* byteVal1 = (byte*)(val1);
    byte donnees[4] = {byteVal0[0], byteVal0[1],
                      byteVal1[0], byteVal1[1]};
    Serial.write(donnees, 4);
}
```

PROGRAMMATION DE L'ORDINATEUR

❖ Choix du langage Python:

- Connaissance acquise en CPGE
- Dispose de librairies pour la lecture sur le port série (PySerial)
- Dispose de tout le nécessaire pour le tracé et son animation avec matplotlib

```
# Création de la figure
fig = plt.figure(figsize=(10, 8))
# Ajout des axes à la fiugure
ax = plt.axes(xlim=(xmin, xmax), ylim=(ymin, ymax))
ax.set_title('Signaux A0 et A1')                                # Titre
ax.set_xlabel("Temps")                                         # Label axe X
ax.set_ylabel("Sorties A0 et A1 (V)")                           # Label Axe Y
lineLabel = ['Signal Traité', 'Signal brut']                   # Légende des courbes
style = ['r-', 'c-']                                            # Styles de ligne pour les deux tracés
lines = []                                                       # Tableau des tracés de lignes
for i in range(2):                                              # Ajout des deux tracés de ligne au tableau
    # Axes.plot renvoie une liste de lignes représentant les tracés effectués => ici un seul par appel, d'où [0]
    lines.append(ax.plot([], [], style[i], label=lineLabel[i])[0])
# Lancement de l'animation : le second argument est la fonction à appeler pour mettre à jour le 3e argument
# fargs doit absolument etre un tuple
anim = animation.FuncAnimation(fig, capteur.alimenterDonneesAnimation, fargs=(lines,), interval=pltInterval)
```



ANNEXE 3

Théorie des filtres numériques

PRÉAMBULE : TRANSFORMÉE DE FOURIER DISCRÈTE

On considère $x(t)$ un signal continu, échantillonné à la fréquence f_e

- ❖ On appelle *signal discret associé* la suite $(x_n)_{n \in \mathbb{Z}}$ telle que
 $x_n = x(n \cdot T_e)$ (avec $T_e = \frac{1}{f_e}$)
- ❖ On appelle *transformée de Fourier discrète* l'expression
$$\hat{X}(f) = \sum_{n=-\infty}^{n=+\infty} x_n e^{-i2\pi f n T_e}$$
- ❖ En posant $Z = e^{i2\pi f T_e}$, on obtient *la transformée en Z du signal discret*

$$X(Z) = \sum_{n=-\infty}^{n=+\infty} x_n Z^{-n}$$

FILTRAGE NUMÉRIQUE LINÉAIRE – STABILITÉ D'UN FILTRE

Soit x_n un signal numérique d'entrée échantillonné à la fréquence f_e .

- ❖ Le filtrage numérique linéaire consiste à obtenir un nouveau signal numérique y_n en utilisant une relation de récurrence de la forme suivante :

$$y_n = \sum_{k=0}^{n-1} a_k x_{n-k} + \sum_{k=1}^{n-1} b_k y_{n-k} \text{ avec } (a_k)_{k \in \mathbb{N}} \text{ et } (b_k)_{k \in \mathbb{N}} \text{ suites réelles}$$

- ❖ Ainsi, d'une manière très générale, on obtient l'échantillon n du signal de sortie par une combinaison linéaire de n échantillons précédents du signal d'entrée et de $n-1$ échantillons précédents du signal de sortie (déjà calculés).
- ❖ On dit qu'un filtre est *stable* si $\lim_{n \rightarrow +\infty} y_n = 0$

RÉPONSE IMPULSIONNELLE

- ❖ On appelle *impulsion* un signal discret $(x_n)_{n \in \mathbb{N}}$ défini par :

$$\begin{cases} x_0 = 1 \\ x_n = 0 \text{ pour } n > 0 \end{cases}$$

- ❖ On appelle *réponse impulsionnelle* d'un filtre la suite $(y_n)_{n \in \mathbb{N}}$ constituant le signal de sortie calculé à partir de l'impulsion
- ❖ La réponse impulsionnelle est dite *finie* si la suite $(y_n)_{n \in \mathbb{N}}$ est presque nulle
- ❖ Par conséquent, *un filtre à réponse impulsionnelle finie est forcément stable*

SYSTÈME CAUSAL À RÉPONSE IMPULSIONNELLE FINIE

- ❖ Si l'échantillon n du signal de sortie est une combinaison linéaire exclusive des échantillons 0 à $n-1$ du signal d'entrée, alors le système est dit *causal*.

$$y_n = \sum_{k=0}^{n-1} h_k x_{n-k}$$

- ❖ La réponse à l'impulsion est alors $y_n = h_n$, et on appelle ainsi $(h_n)_{n \in \mathbb{N}}$ la réponse impulsionnelle du système.
- ❖ Si $(h_n)_{n \in \mathbb{N}}$ est presque nulle, nous avons défini un filtre causal à réponse impulsionnelle finie (RIF), qui est par définition stable. Nous allons utiliser ces filtres dans la suite de nos travaux.

RÉPONSE FRÉQUENTIELLE D'UN FILTRE RIF

- ❖ Supposons une entrée $x_n = e^{i2\pi f n T_e}$ sinusoïdale de fréquence f d'amplitude 1

$$y_n = \sum_{k=0}^{N-1} h_k x_{n-k} = \sum_{k=0}^{N-1} h_k e^{i2\pi f(n-k)T_e}$$

$$= e^{i2\pi f n T_e} \sum_{k=0}^{N-1} h_k e^{-i2\pi f k T_e} = \hat{H}(f) x_n$$

Ainsi, la réponse fréquentielle du filtre est :

$$\hat{H}(f) = \sum_{k=0}^{N-1} h_k e^{-i2\pi f k T_e}$$

ANALYSE D'UN FILTRE NUMÉRIQUE PASSE BAS(1/2)

- ❖ Dans la bande passante de notre filtre, le signal ne doit pas être déformé. Ainsi, toutes les fréquences de la BP doivent subir un retard constant τ .
- ❖ Supposons une composante de fréquence f dans la bande passante: $x_n = e^{i2\pi f n T_e}$

On veut, en posant $\tau = P T_e$, avec P entier

$$y(n T_e) = x(n T_e - P T_e) \text{ (Gain de 1), soit } y_n = x_{n-P}$$

$$\sum_{k=0}^{N-1} h_k e^{i2\pi f(n-k) T_e} = e^{i2\pi f(n-P) T_e} \text{ soit } \sum_{k=0}^{N-1} h_k e^{-i2\pi f k T_e} = e^{-P i2\pi f P T_e}$$

$$\widehat{H}(f) = e^{-i2\pi f P T_e}$$

=> Gain de 1, déphasage proportionnel à la fréquence.

ANALYSE D'UN FILTRE NUMÉRIQUE PASSE BAS(2/2)

❖ Dans la bande passante :

$$\widehat{H}(f) = \sum_{k=0}^{N-1} h_k e^{-i2\pi f k T_e} = e^{-i2\pi f P T_e}$$

$$\sum_{k=0}^{N-1} h_k e^{-i2\pi f(k+P) T_e} = 1 = G(f)$$

$$G(f) = 1 = \sum_{k=-P}^{N-1-P} h_{k+P} e^{-i2\pi f k T_e}$$

❖ G est réel donc $h_{P-k} = h_{P+k}$ et N est impair = $2P+1$

SYNTHESE D'UN FILTRE NUMERIQUE PASSE BAS(1/3)

❖ Soit un filtre répondant aux hypothèses de l'analyse

Posons $g_k = h_{k+P}$ pour $-P \leq k \leq P$

$$G(f) = \sum_{k=-P}^{N-1-P} h_{k+P} e^{-i2\pi fkT_e} = \sum_{k=-P}^P g_k e^{-i2\pi fkT_e}$$

❖ G est donc la transformée de Fourier discrète de (g_n) . Ainsi comme G est périodique de période f_e , alors les termes g_k seront les coefficients de Fourier de la fonction de gain G.

$$g_k = \frac{1}{f_e} \int_{-\frac{f_e}{2}}^{\frac{f_e}{2}} G(f) e^{i2\pi fkT_e} df = \frac{1}{f_e} \int_{-\frac{f_e}{2}}^{\frac{f_e}{2}} G(f) e^{i2\pi k \frac{f}{f_e}} df$$

SYNTHESE D'UN FILTRE NUMERIQUE PASSE BAS(2/3)

- ❖ Soit un filtre passe bas, de fréquence de coupure f_c , défini sur $\left[-\frac{f_e}{2}, \frac{f_e}{2}\right]$, tel que :

$$\begin{cases} G(f) = 1 \text{ pour } |f| \leq f_c \\ G(f) = 0 \text{ sinon} \end{cases}$$

$$g_k = \frac{1}{f_e} \int_{-\frac{f_e}{2}}^{\frac{f_e}{2}} G(f) e^{i2\pi k \frac{f}{f_e}} df = \frac{1}{f_e} \int_{-f_c}^{f_c} 1 \cdot e^{i2\pi k \frac{f}{f_e}} df$$

$$g_k = \frac{1}{k\pi} \sin\left(2k\pi \frac{f_c}{f_e}\right)$$

- ❖ On déduit de même les coefficients pour les filtres passe haut (signe négatif) et passe bande (somme des g_k)