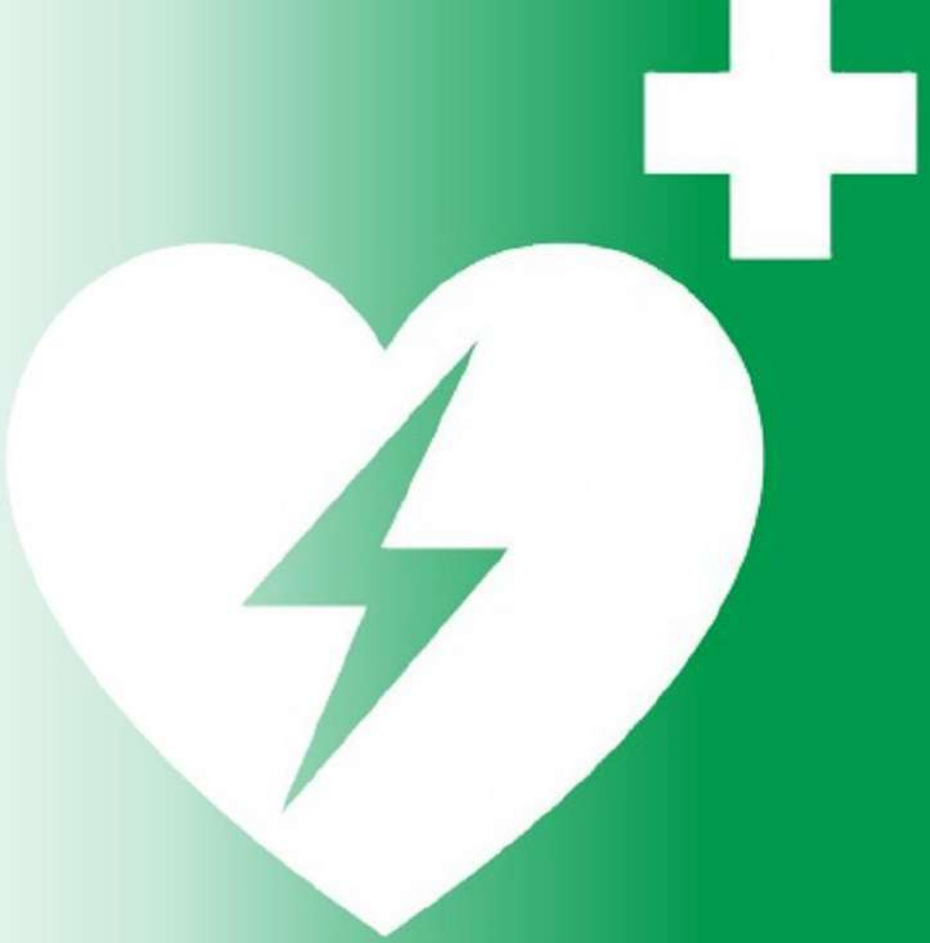
Défibrillateur Automatisé Externe (DAE)



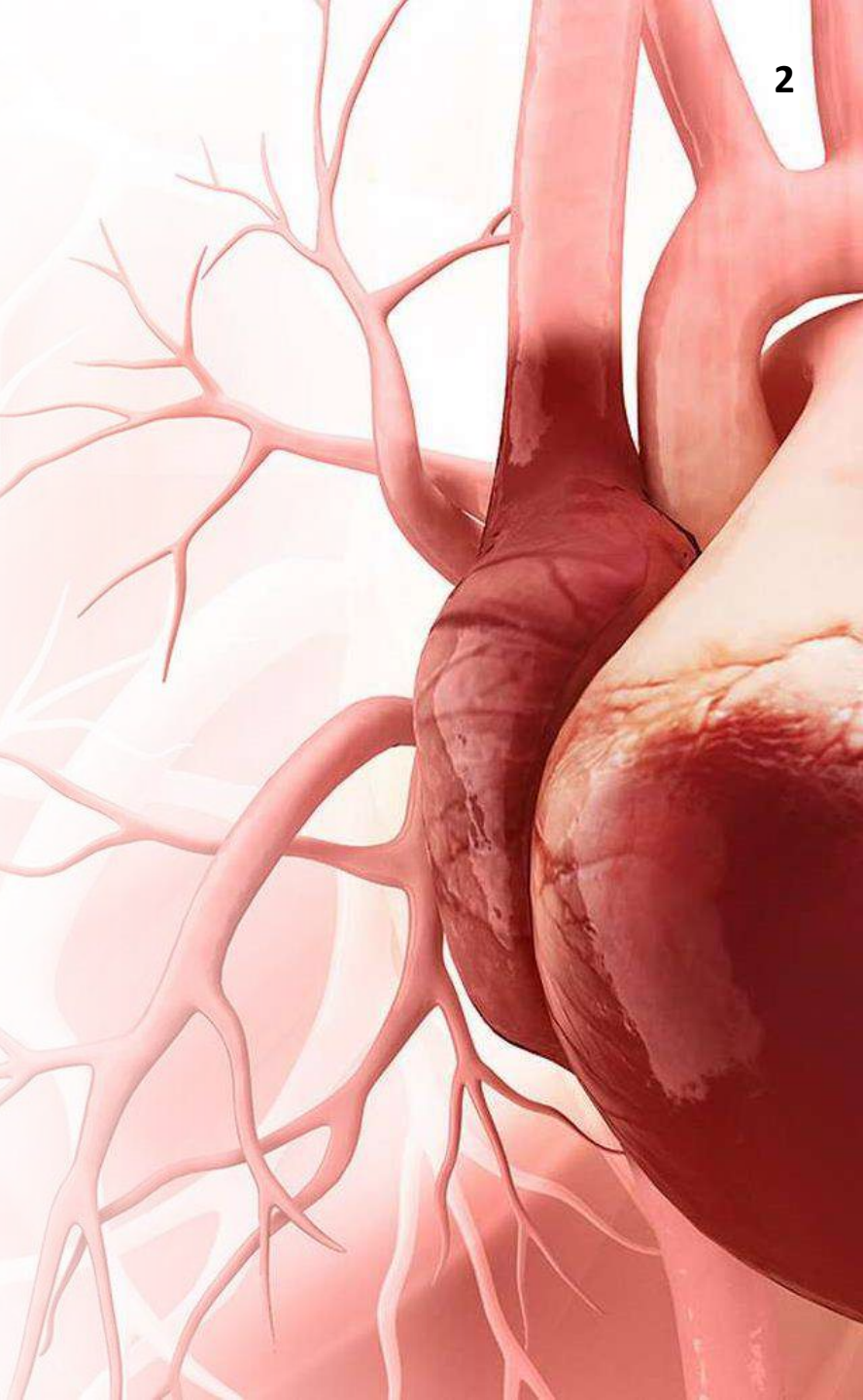
MOREAU Lucie - 47504

Arrêt cardiaque

<u>Nombres</u> :

- ≅ 50000 par an en France
- 75% des cas sont à domicile
- Age moyen : 68 ans

- Taux de survie : 8-10%
- 2ème cause de mortalité en France

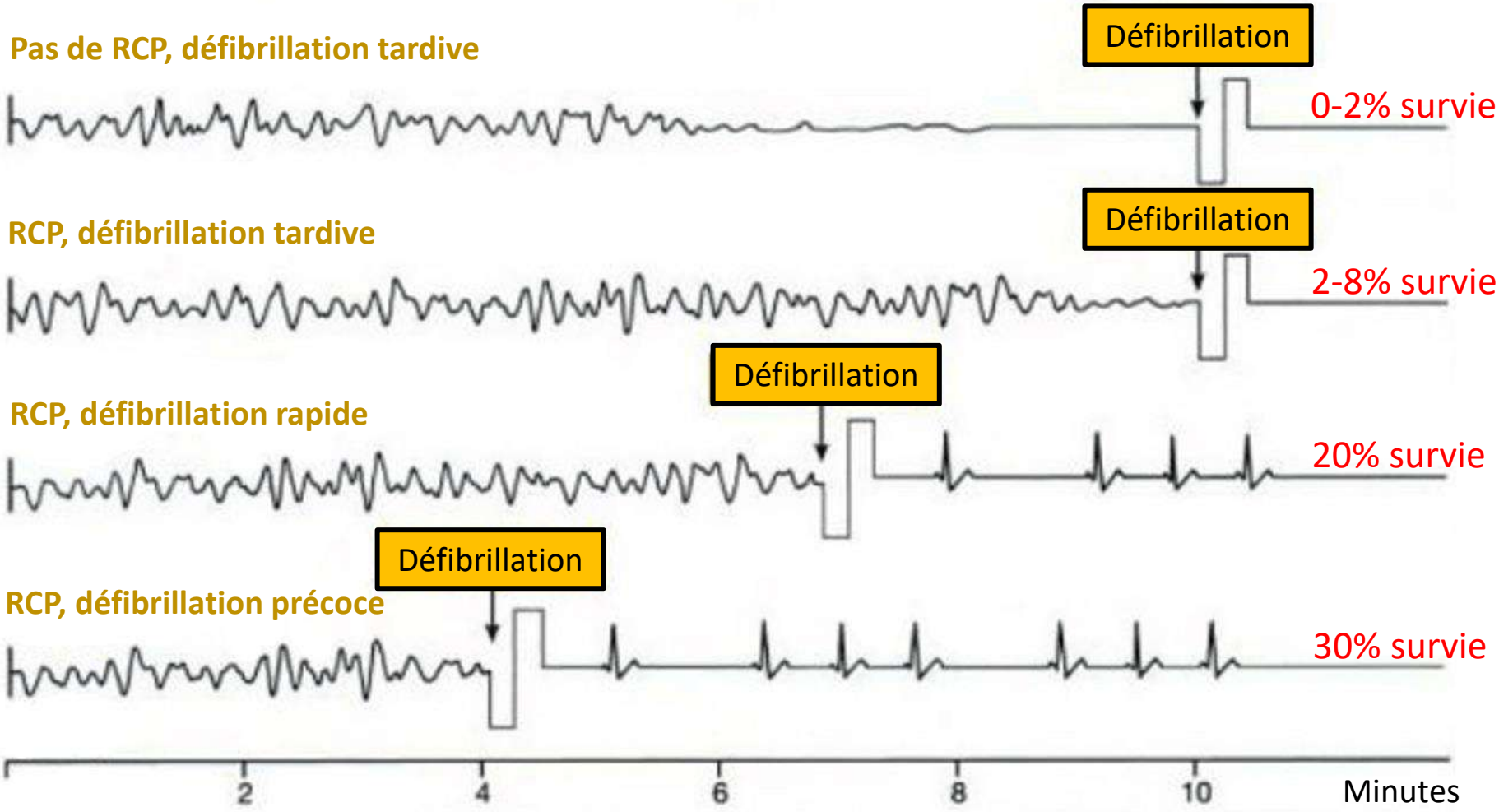


Gestes à accomplir

→ Permet d'augmenter de 4% à 40% les chances de survie de la victime

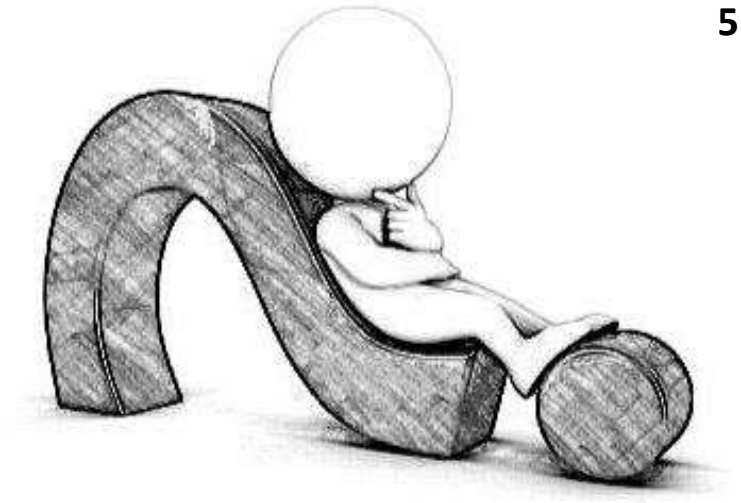
Temps de réaction et efficacité

1 min sans intervention :
chances de survie ↓ 10%



RCP : Réanimation Cardio-Pulmonaire

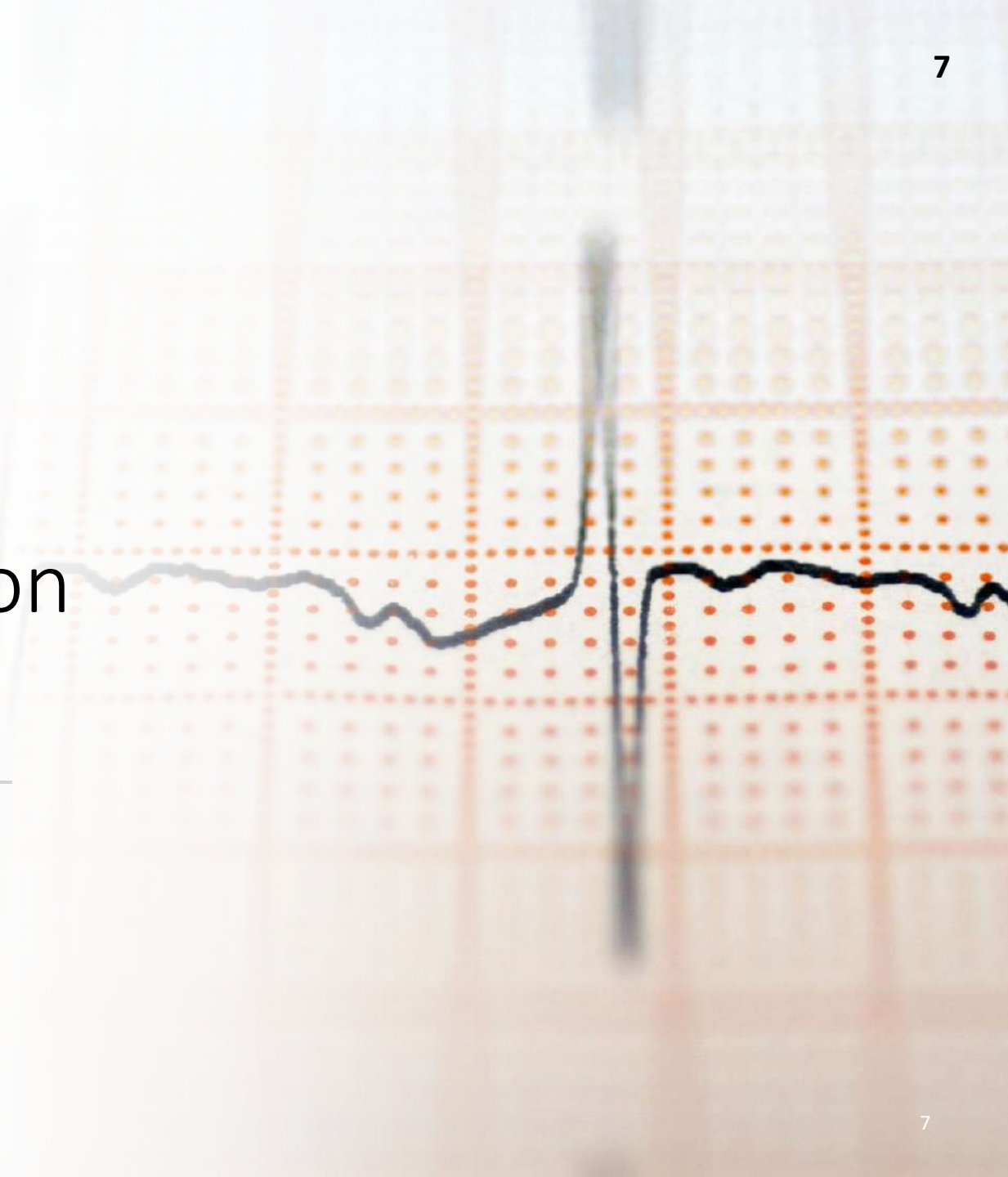
Problématique



De quelle façon le DAE réussit-il à récupérer les données du cœur du patient, les analyser, et les traiter, afin de savoir s'il doit choquer le patient ou non ?

Sommaire

I. Décomposition du DAE



Haut-parleur

Condensateur
2000 V

Isolant galvanique

Carte mère

Blindage

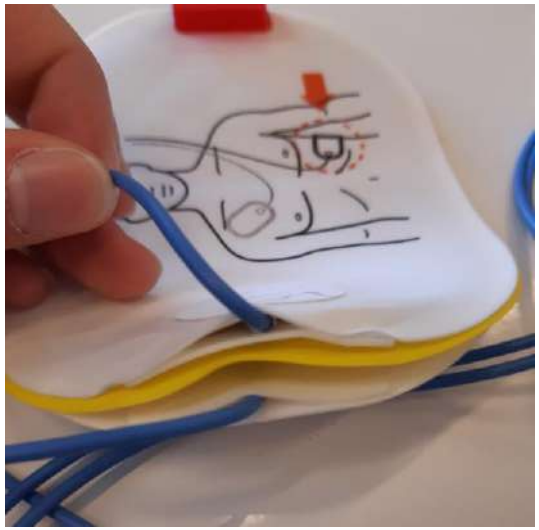
Le condensateur

- $U = 2000V$
- $C = 100\mu F$

$$E = \frac{1}{2} * C * U^2$$

$$E = 200\ J$$

Les électrodes



- Gel :
  → Adaptation d'impédance
- Feuille d'aluminium :
  → Transmission courant

II. Diagramme d'état

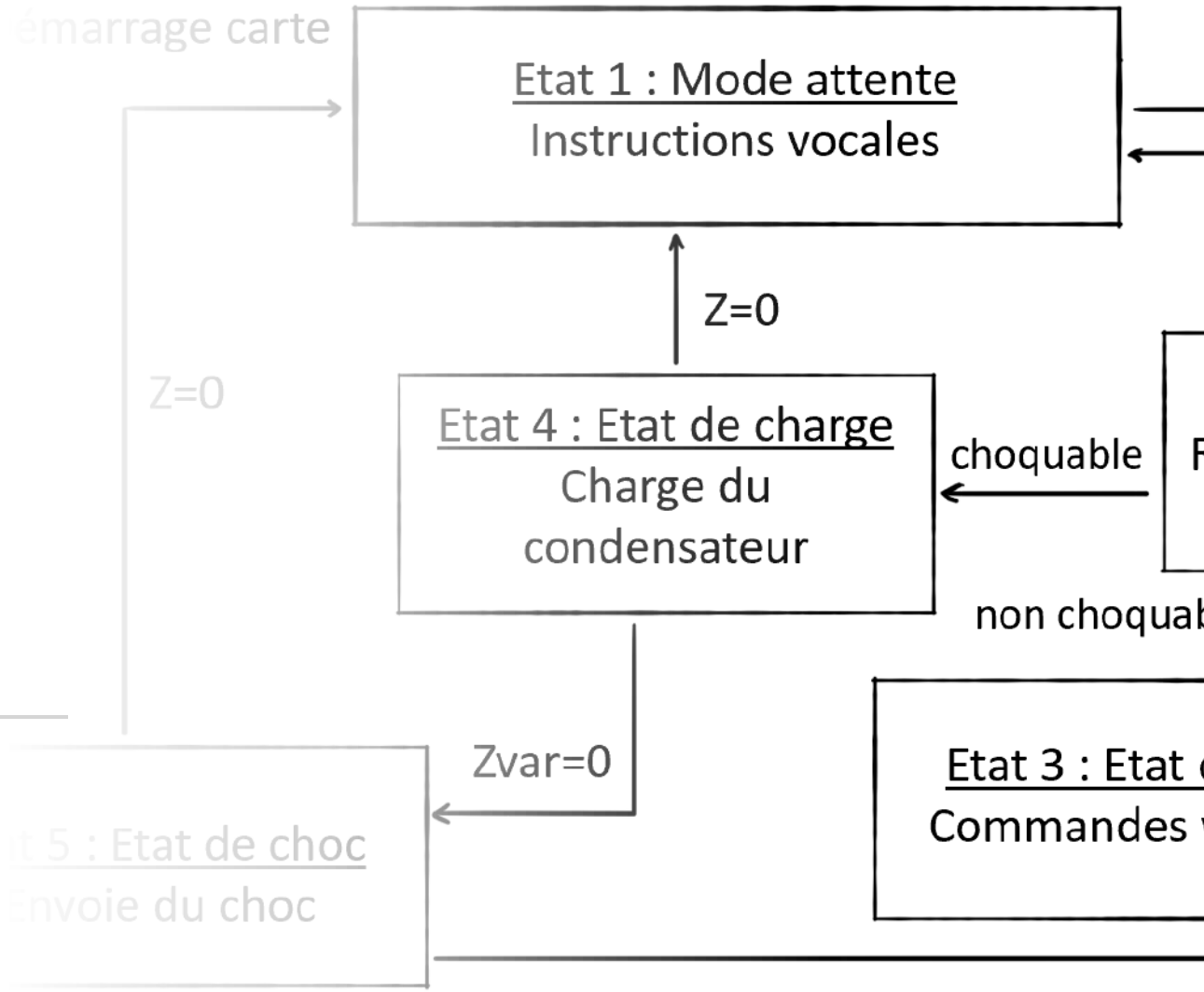
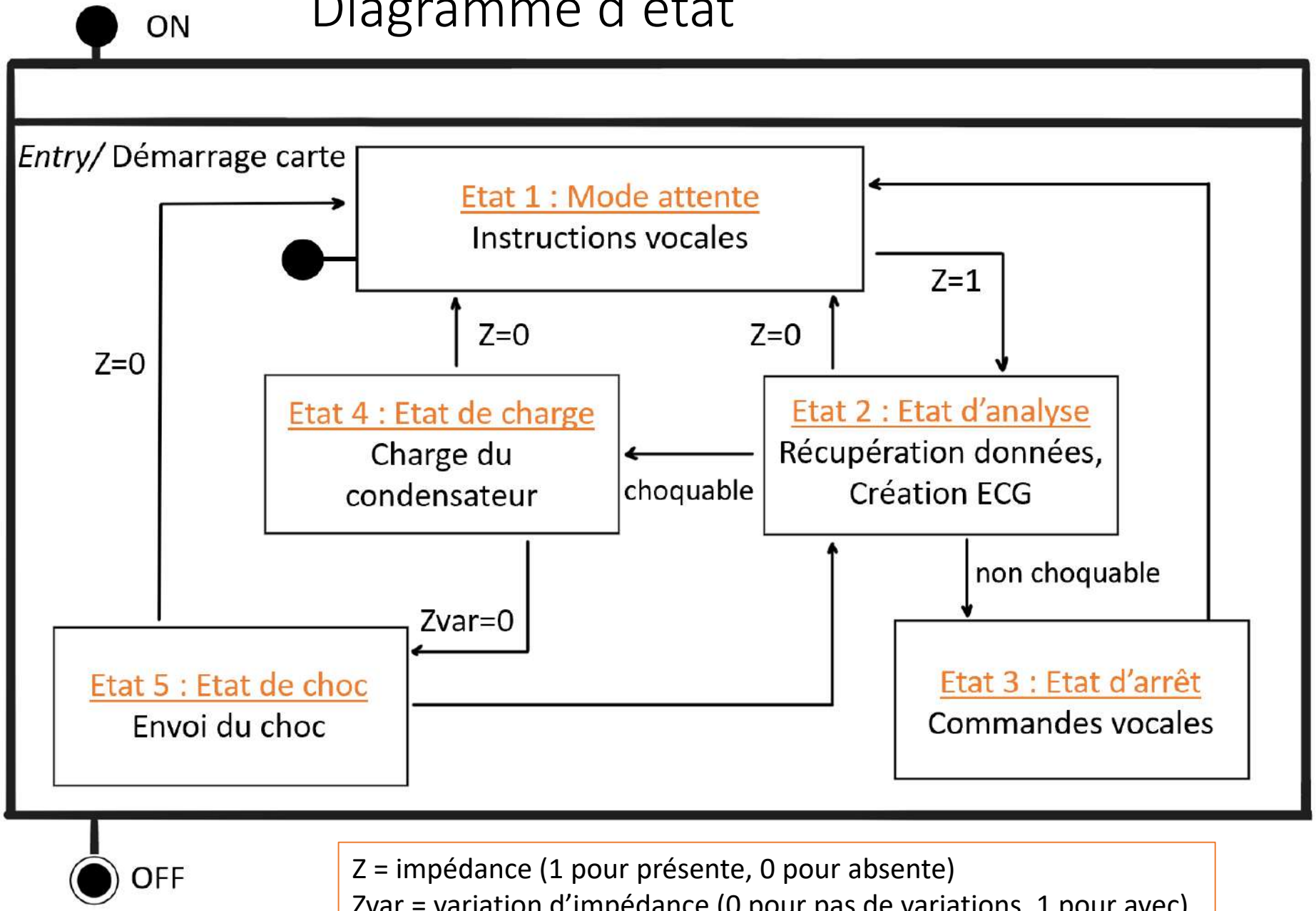
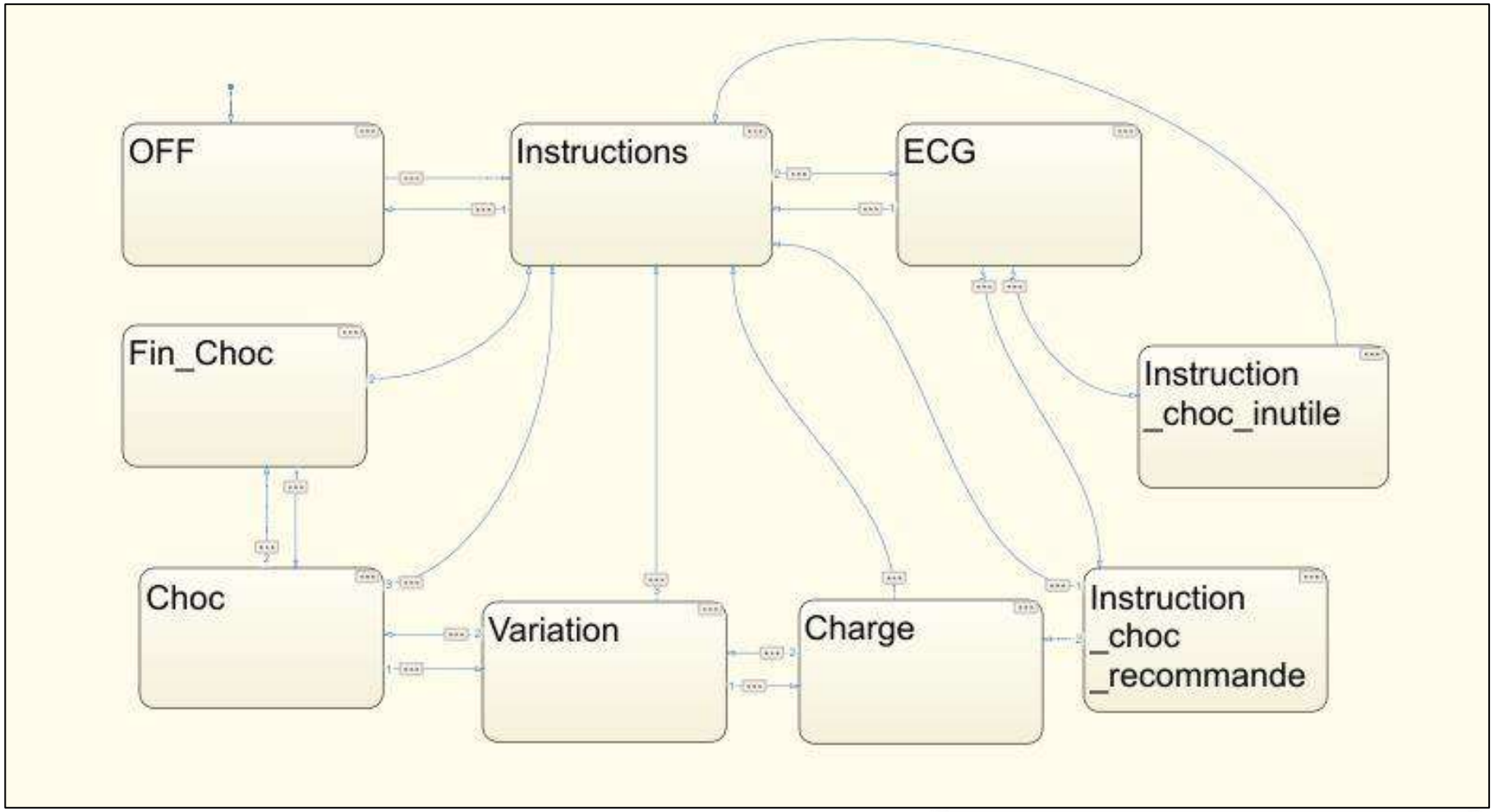


Diagramme d'état

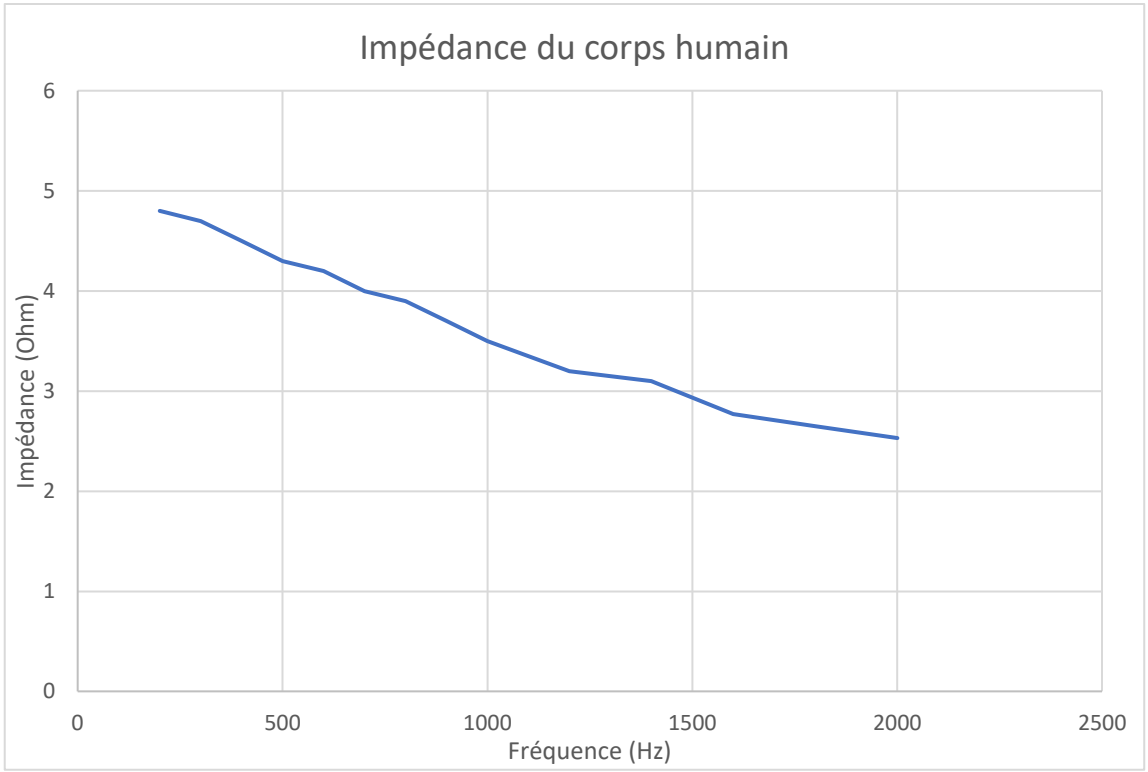


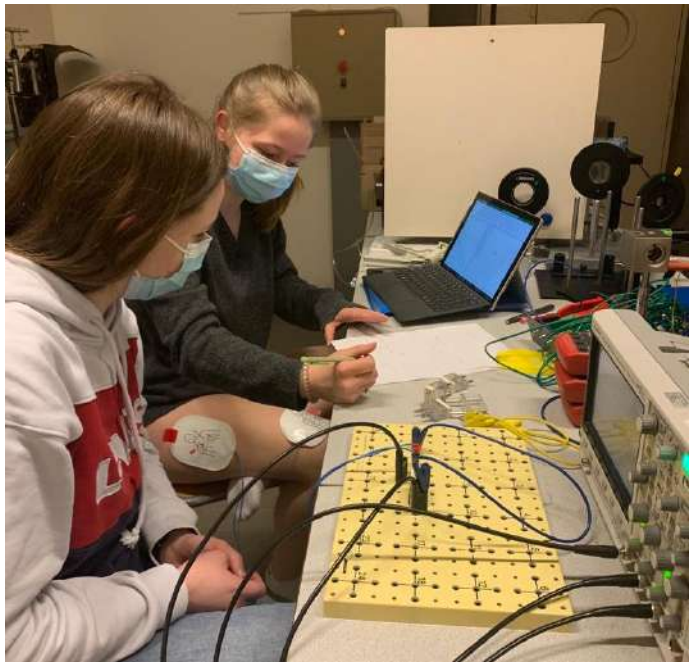
Z = impédance (1 pour présente, 0 pour absente)
Zvar = variation d'impédance (0 pour pas de variations, 1 pour avec)

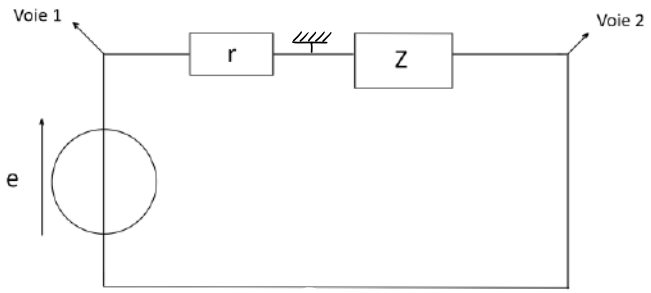
Matlab- Simulink



Impédance - Expérience



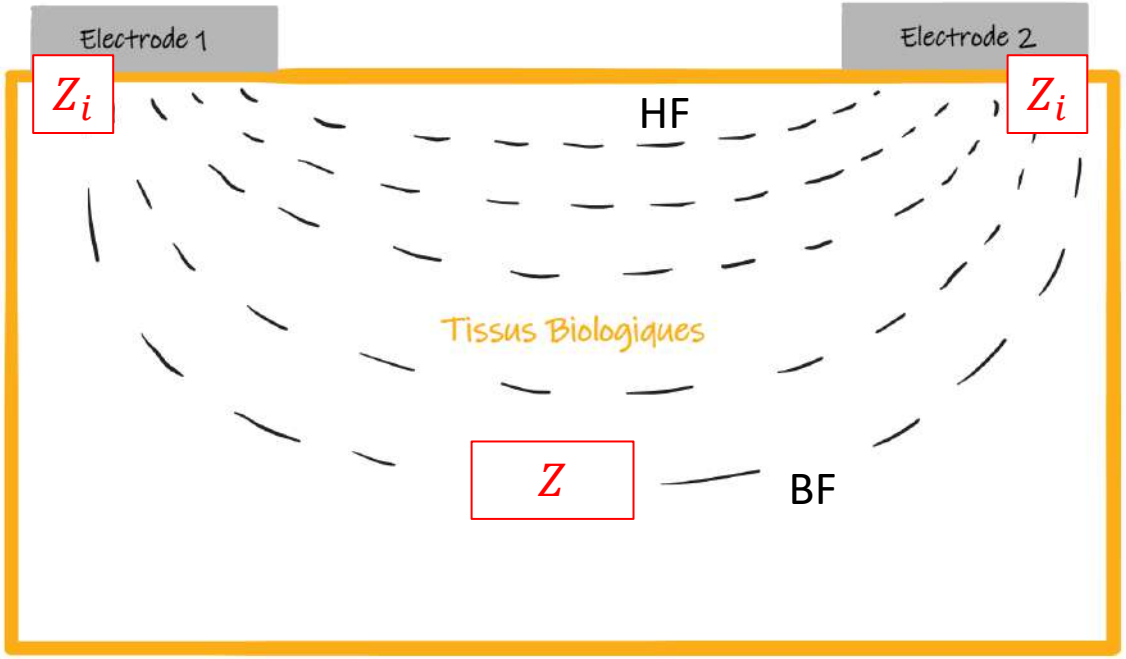




L'impédance diminue avec la fréquence
Pourtant plus le signal est de fréquence élevée, plus il passe en surface.

Impédance – Signal constant 23kHz

Bio-impédance :
Mesure de la résistance des tissus biologiques par l'envoie d'un courant.



Z_i : impédance d'interface Z : impédance thoracique

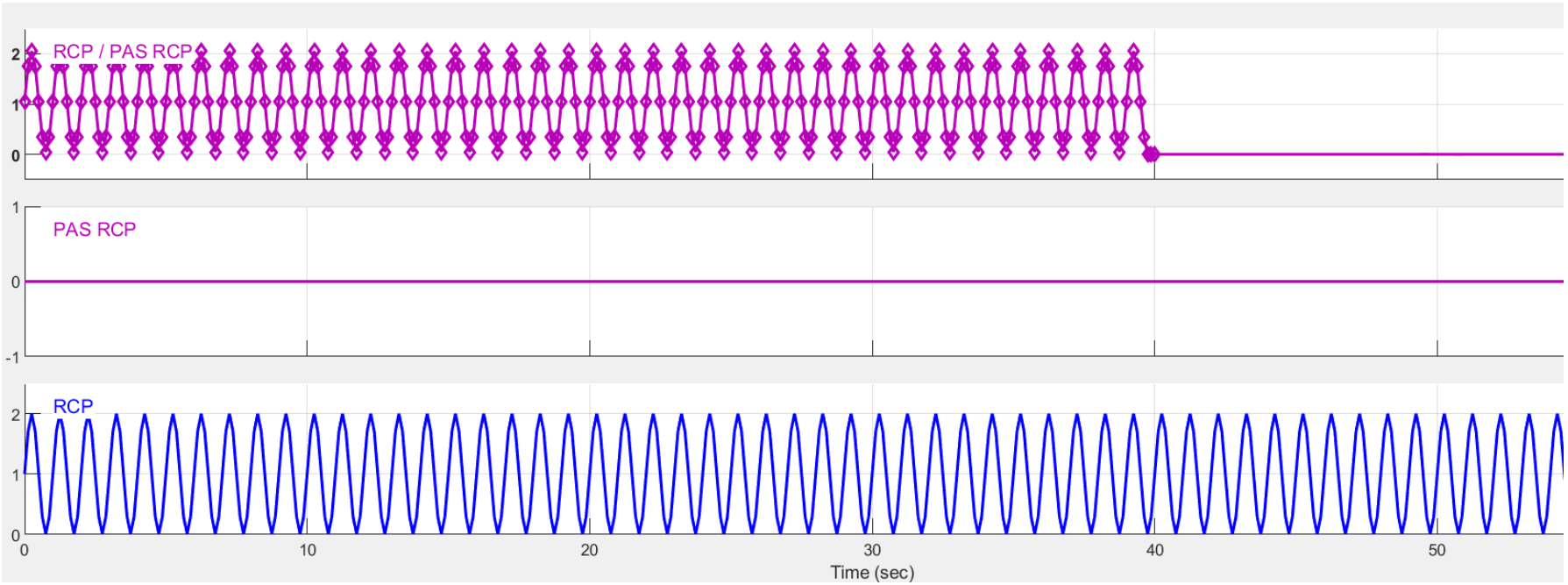
- Signal sinusoidal
- 23kHz
- Faible amplitude

3 impédances en série :

$$Z_i + Z + Z_i$$

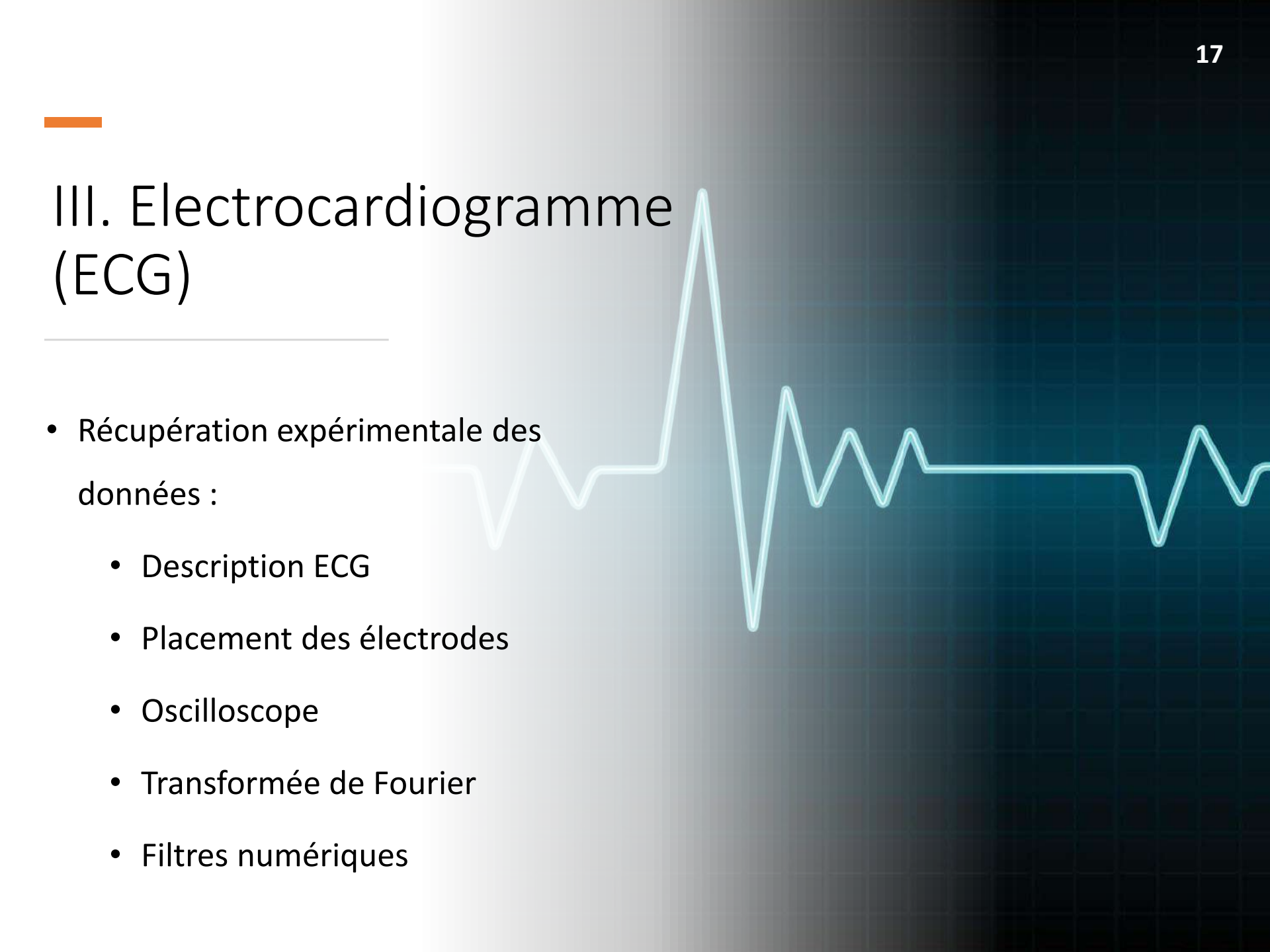
Mais l'impédance d'interface domine !

Variation d'impédance dû au massage cardiaque



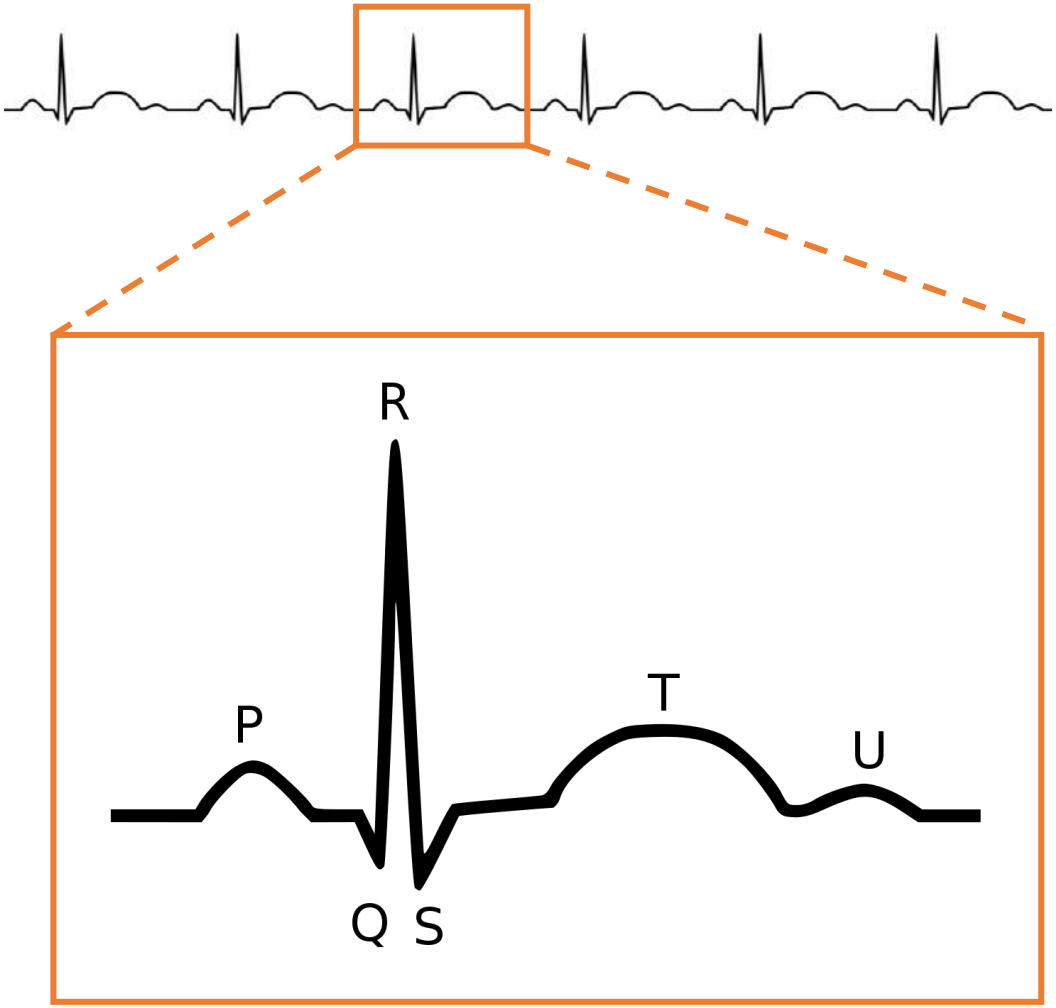
S'illustre par une courbe sinusoïdale de période T=½ s, donc de fréquence 2Hz

III. Electrocardiogramme (ECG)



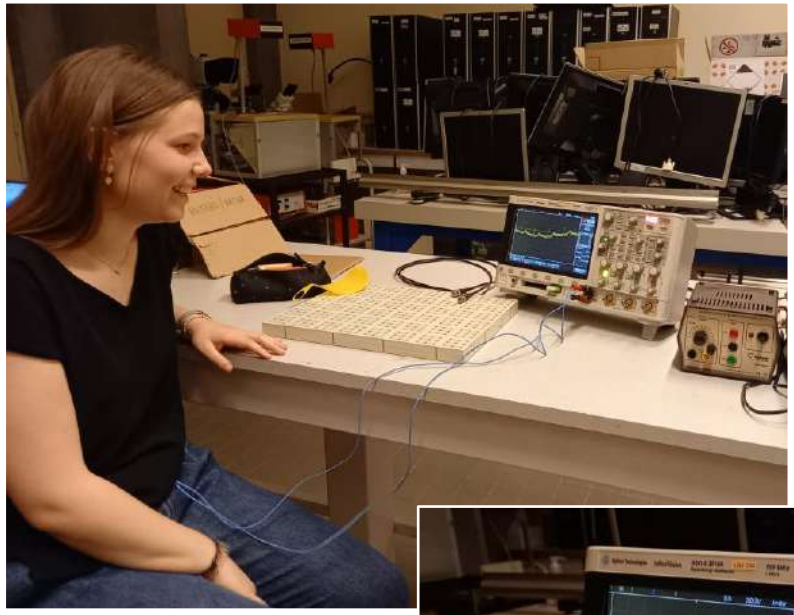
- Récupération expérimentale des données :
  - Description ECG
  - Placement des électrodes
  - Oscilloscope
  - Transformée de Fourier
  - Filtres numériques

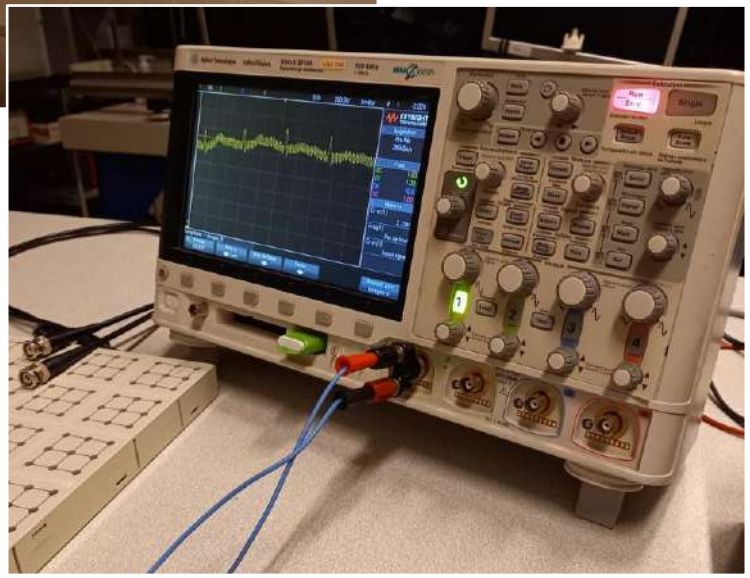
Description ECG



- Onde P : contraction oreillettes
- Complexe QRS : dépolarisation ventricules
- Onde T : repolarisation ventricule
- Onde U : cause incertaine

Récupération expérimentale des données

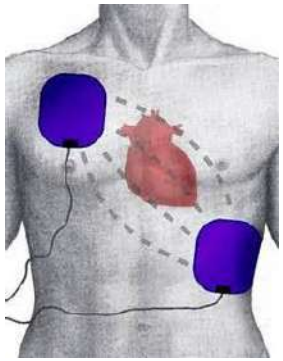


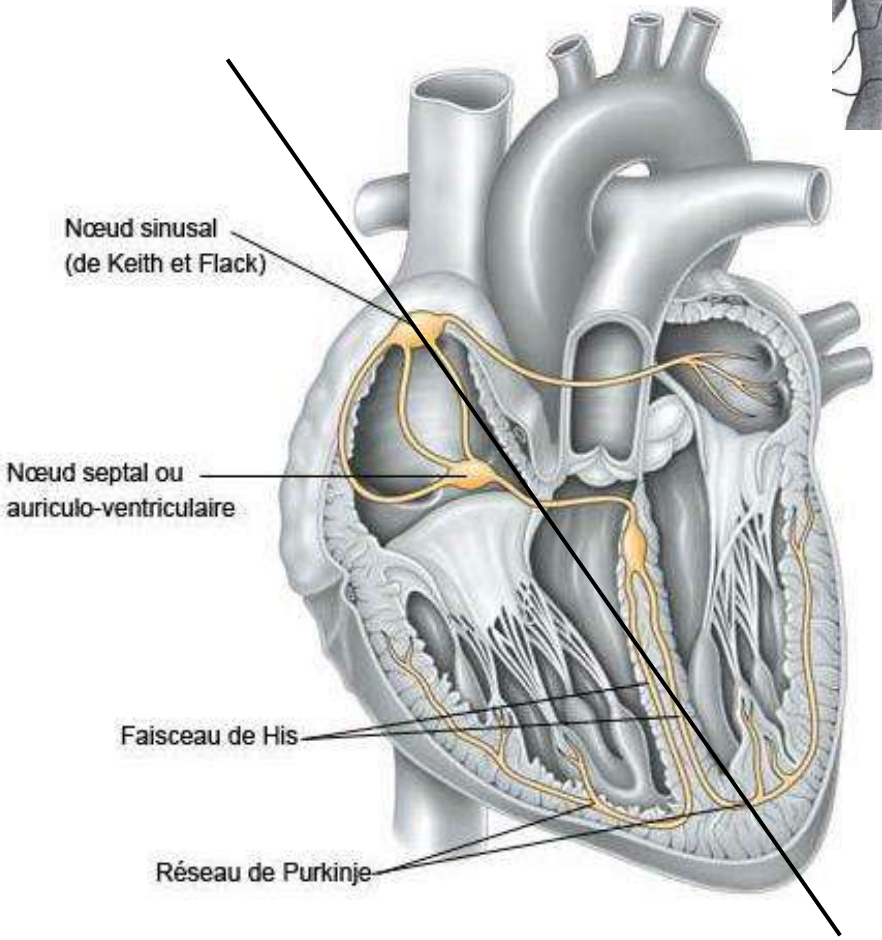


Matériel :

- Electrodes
- Oscilloscope
- Embouts de branchement

Récupération expérimentale des données - Electrodes

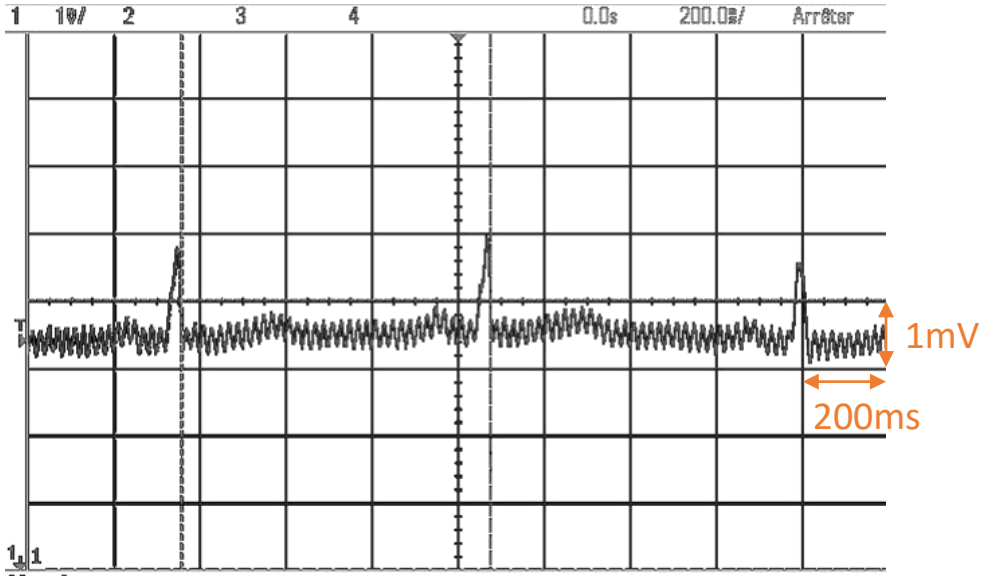


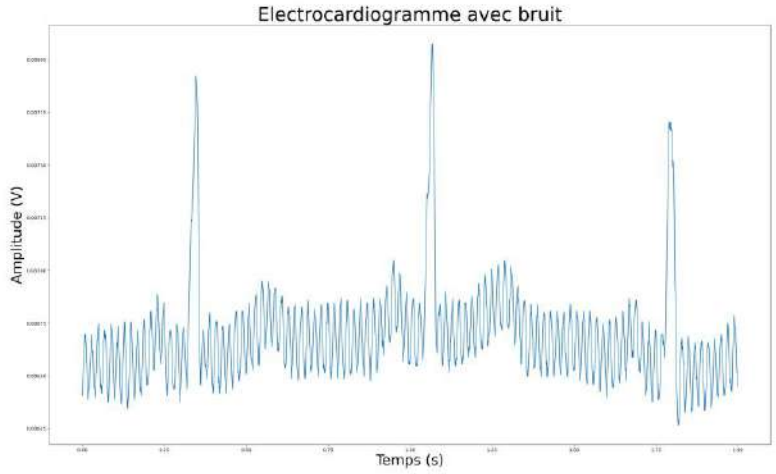


Axe électrique du cœur : direction principale du stimulus électrique

Schéma coupe du cœur vu de face

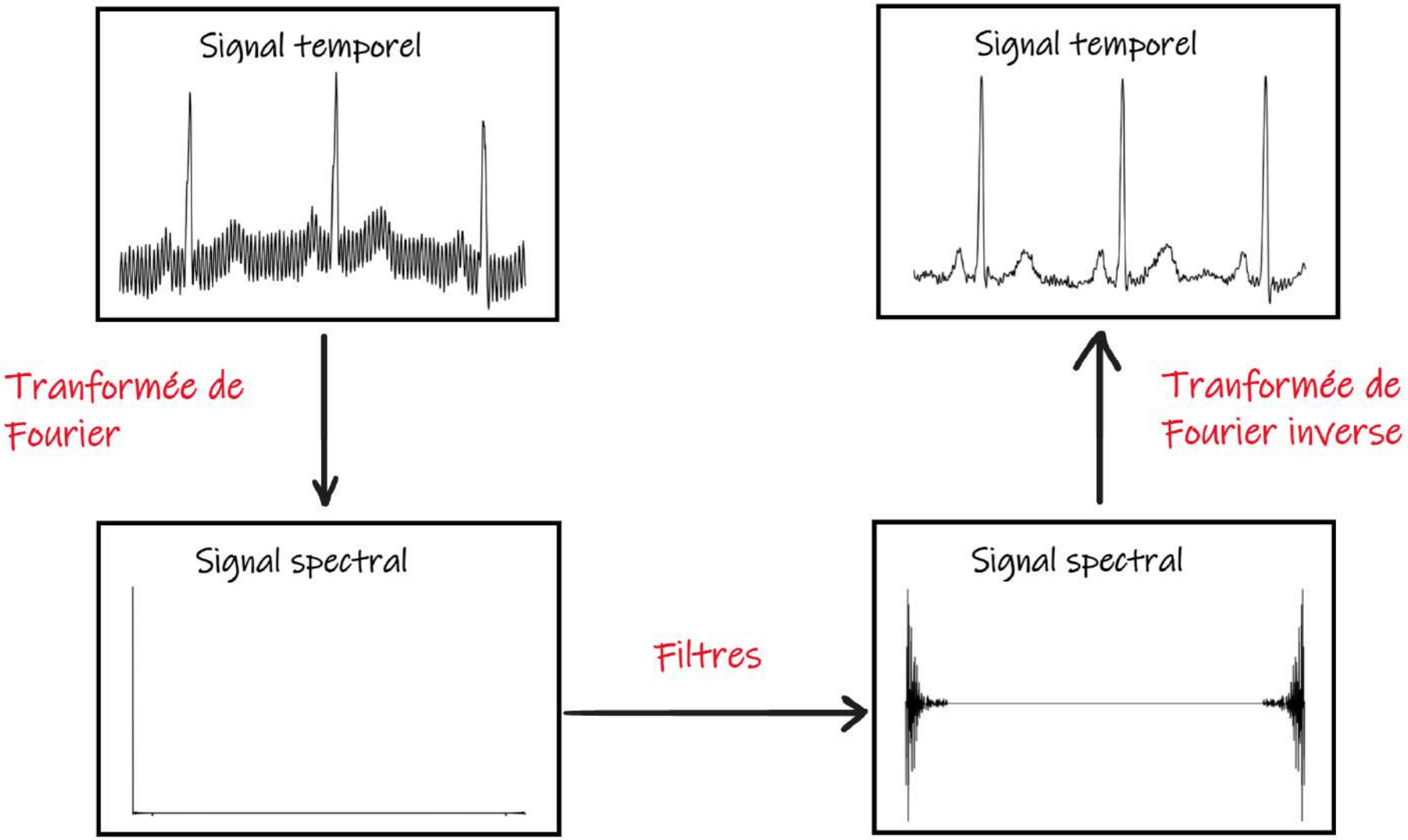
Récupération expérimentale des données - Oscilloscope





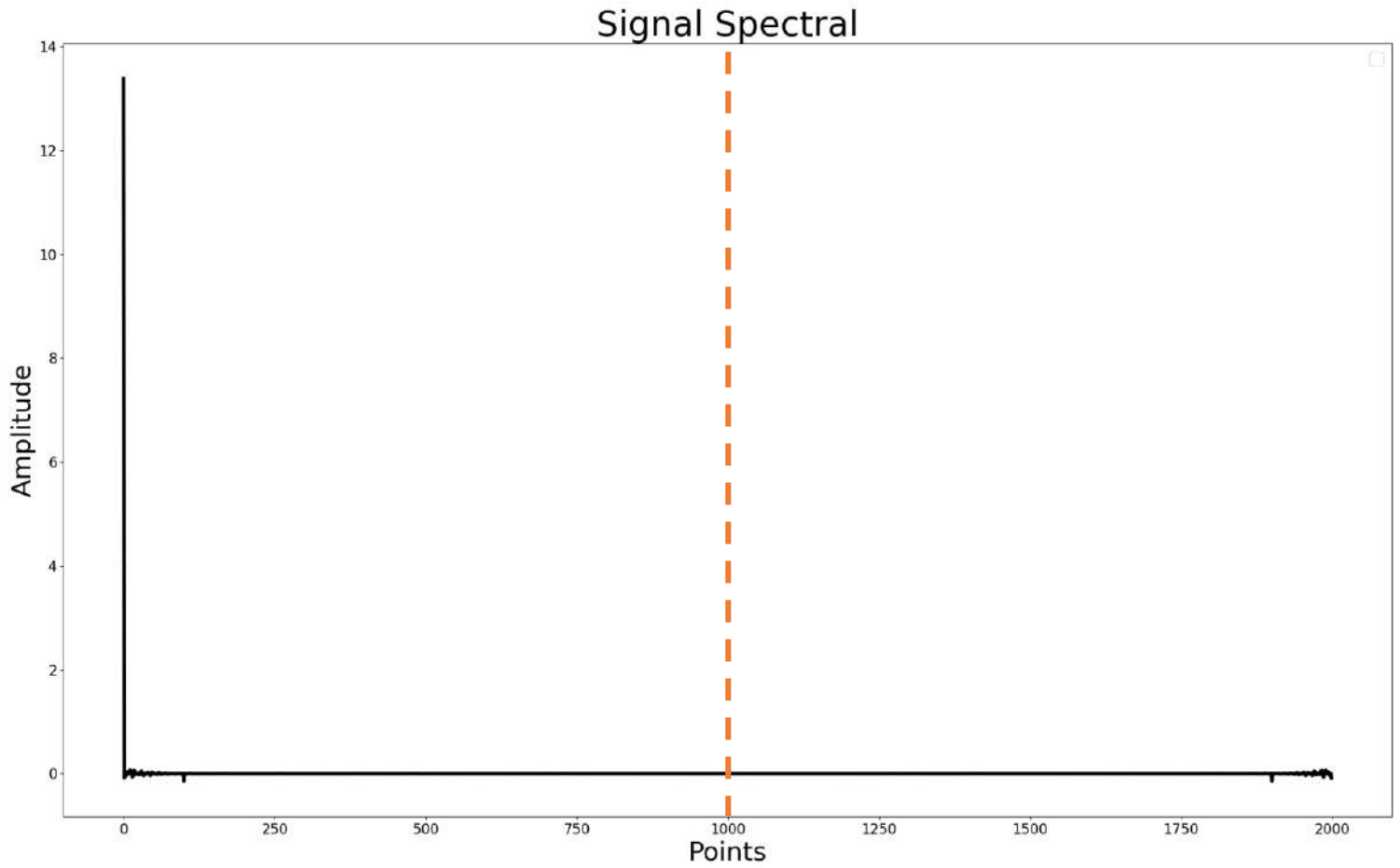
Enregistrement de l'activité électrique du cœur par différence de potentiel

Récupération expérimentale des données – Principe de Filtrage



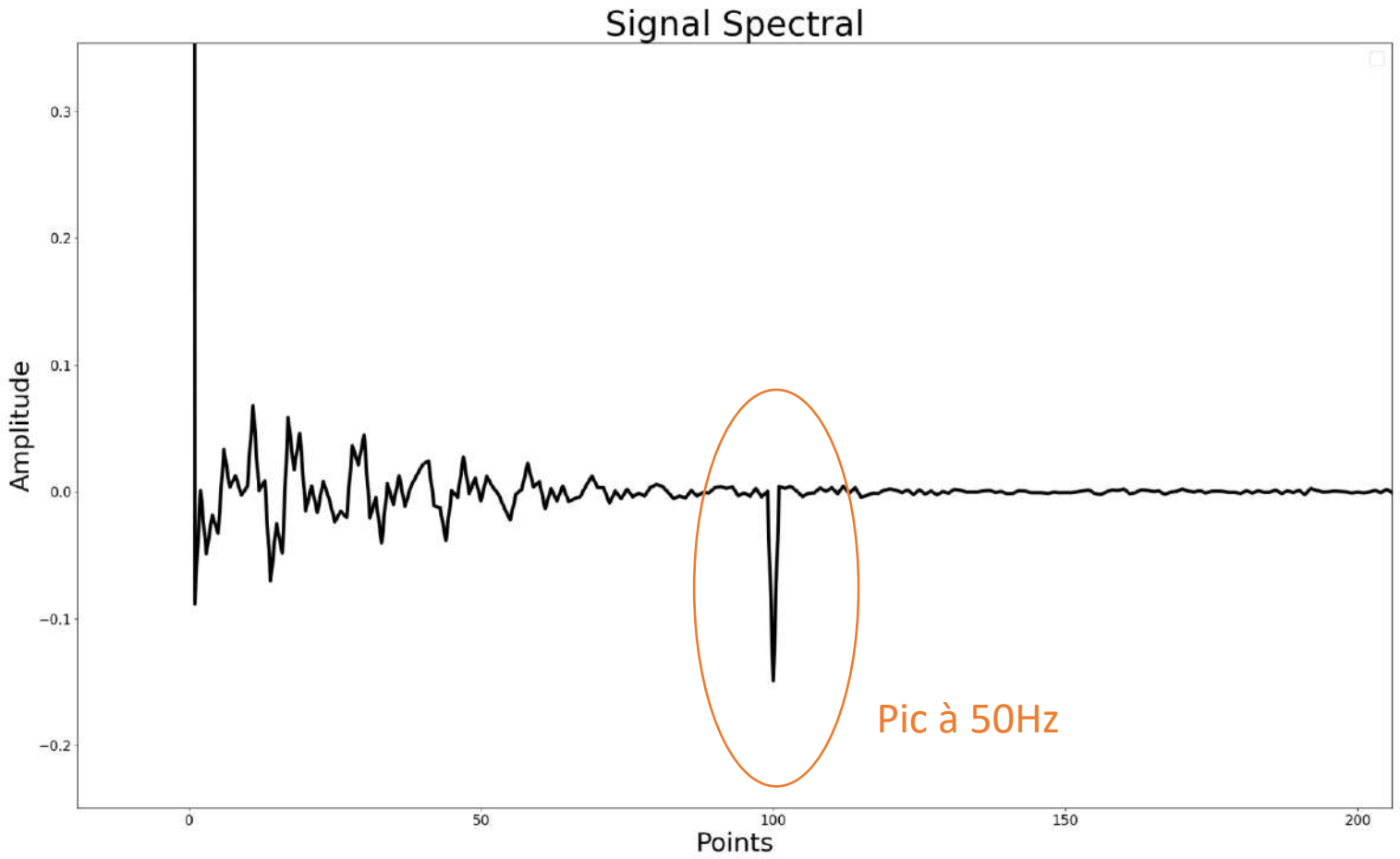
Application de la transformée de Fourier

Symétrie de la transformée avec présence de la moyenne à gauche



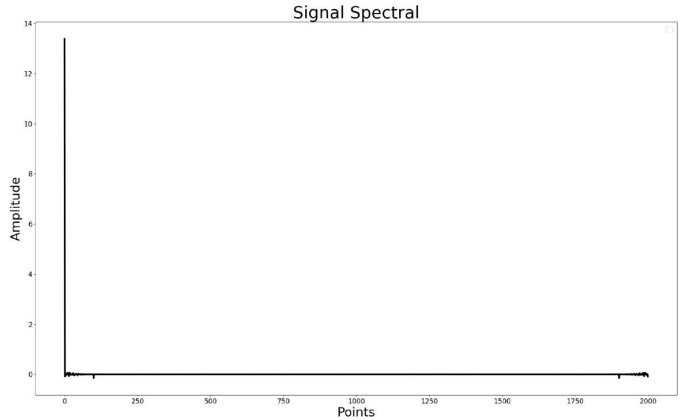
Application de la transformée de Fourier

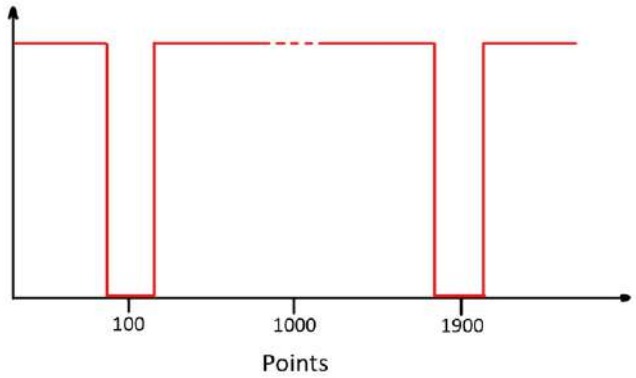
Symétrie de la transformée avec présence du fondamental à gauche

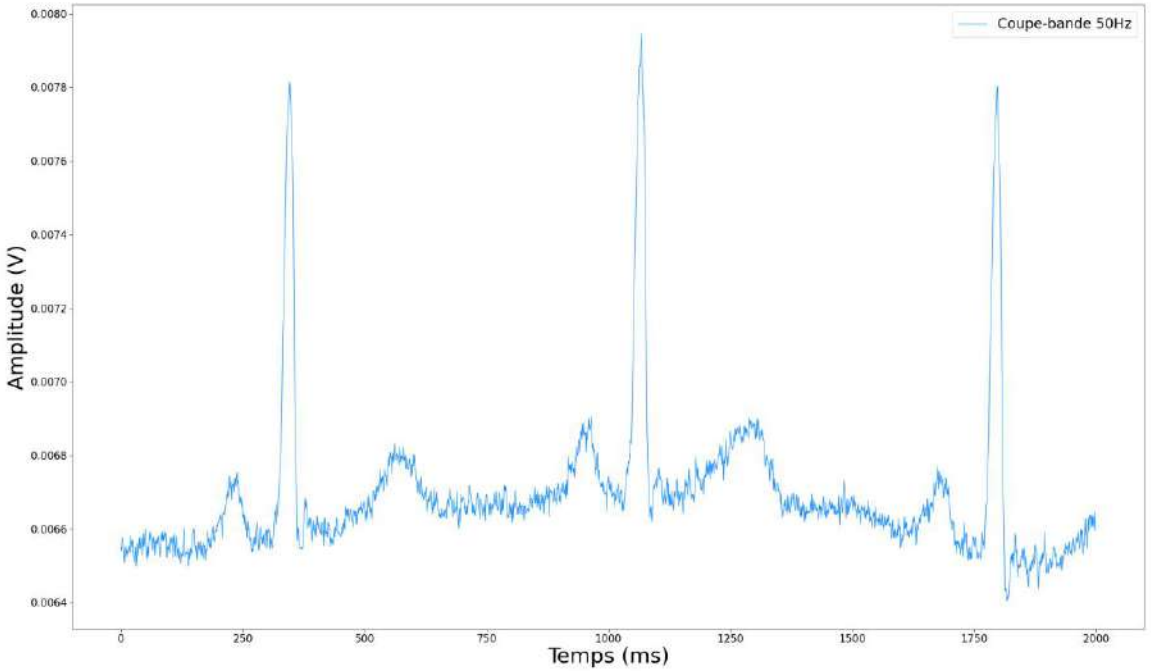


Filtres

Filtre coupe-bande autour de 50Hz (parasitage)

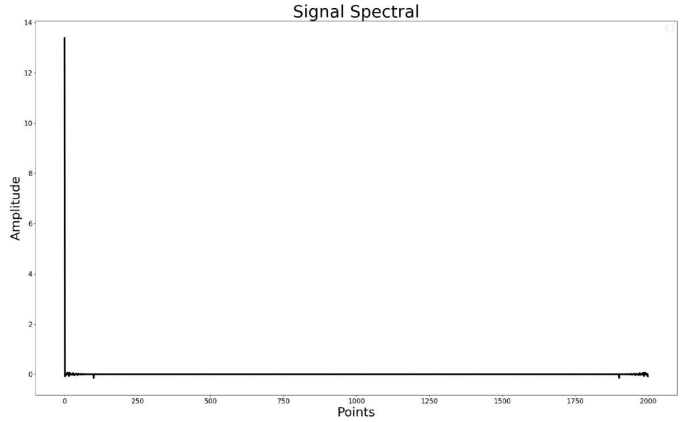


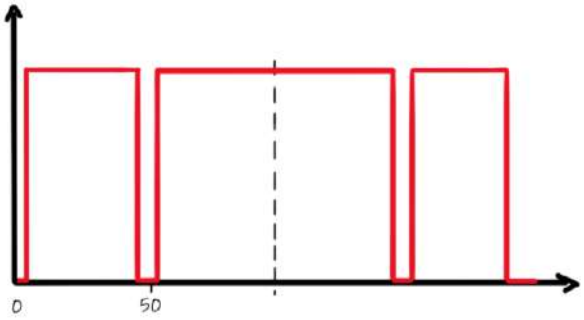


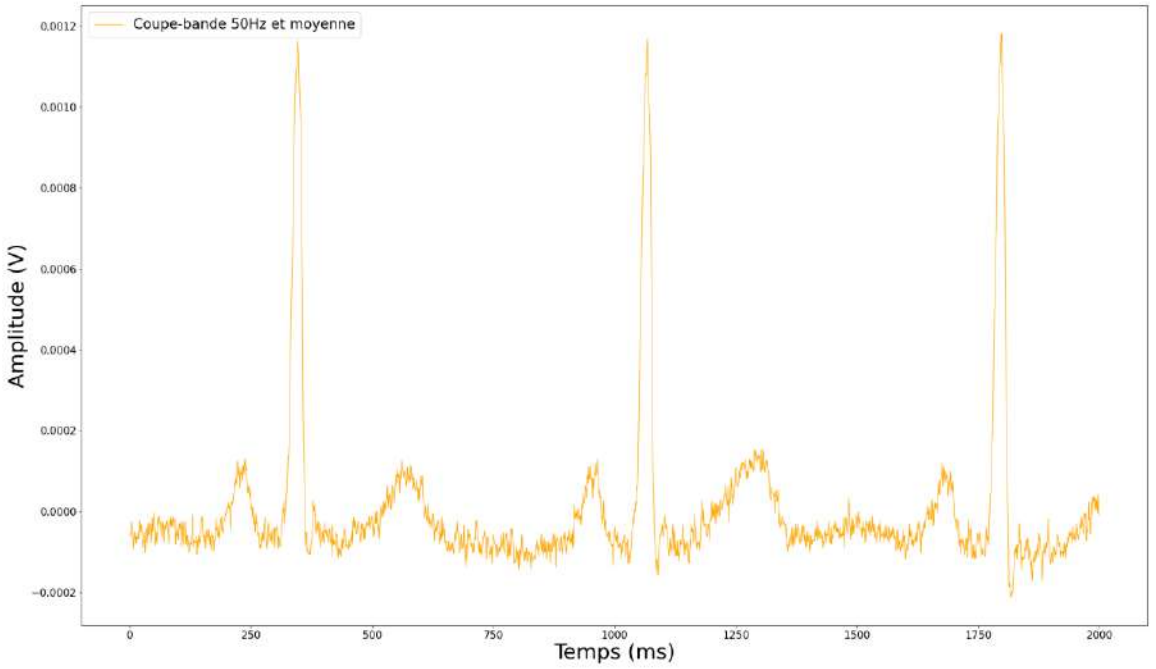


Filtres

+ Filtre coupe-bande de la moyenne



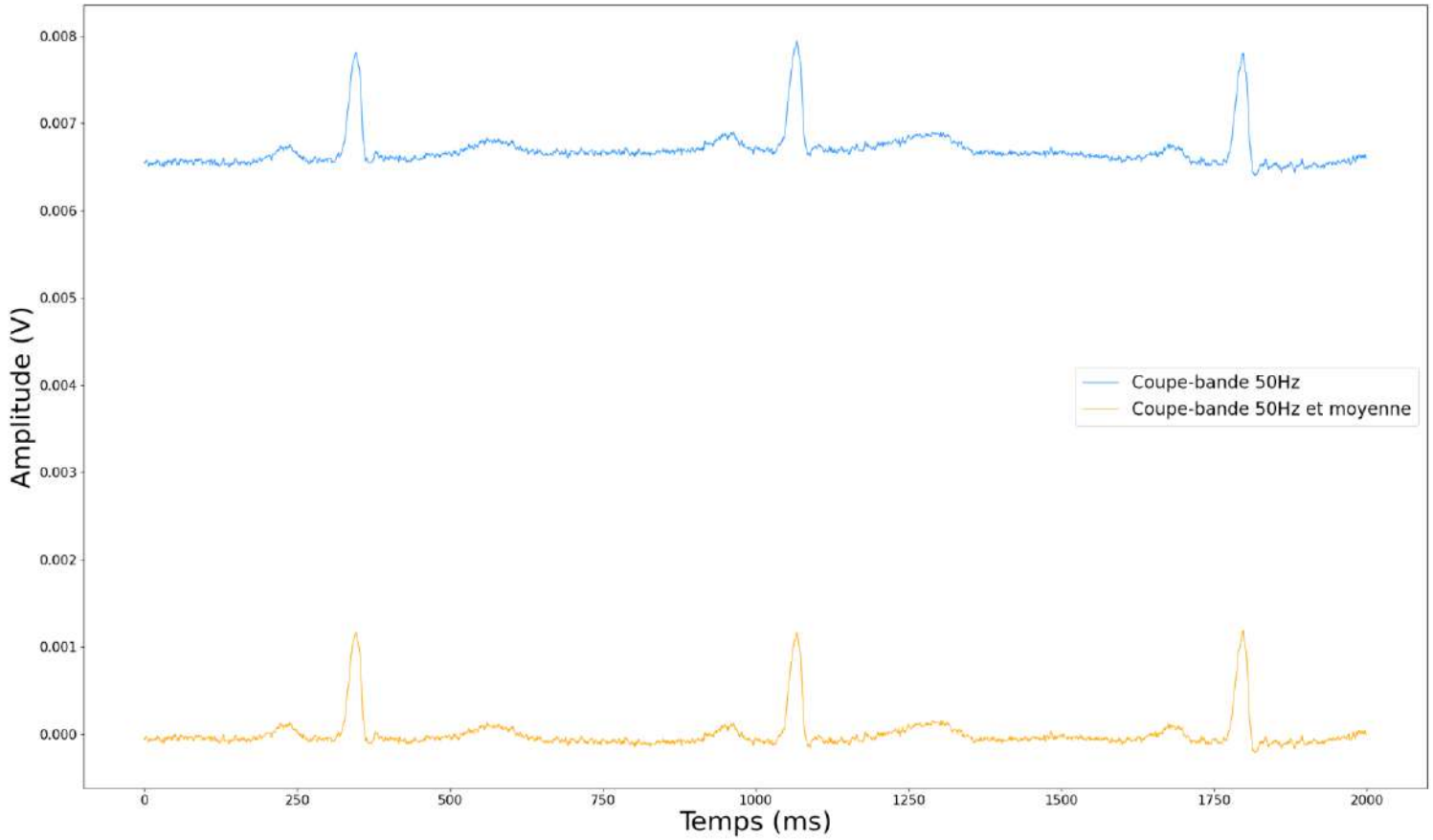




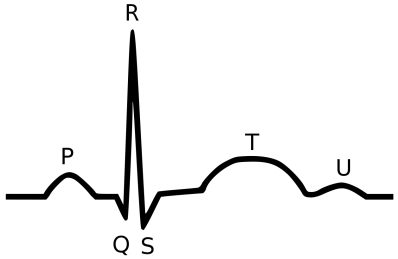
Filtres

Comparaison :

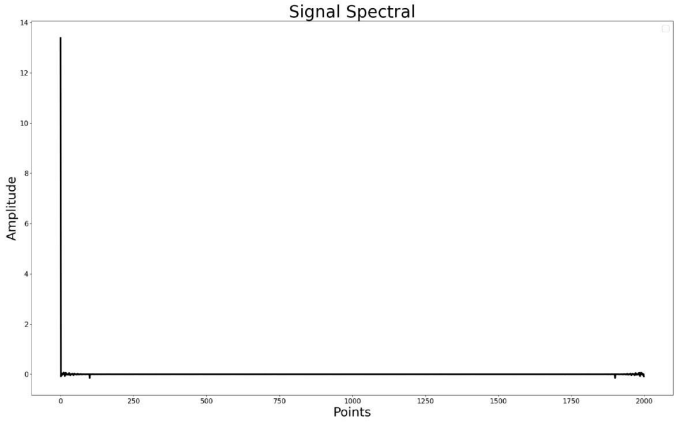
- Filtre coupe-bande 50 Hz
- Filtre coupe-bande 50 Hz et moyenne

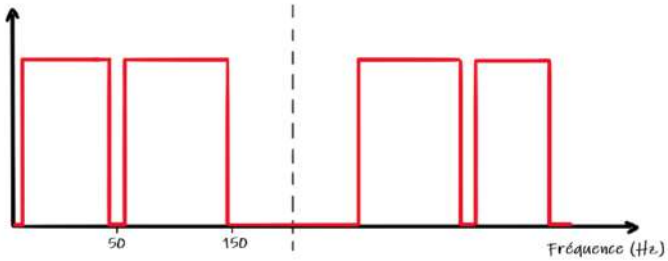


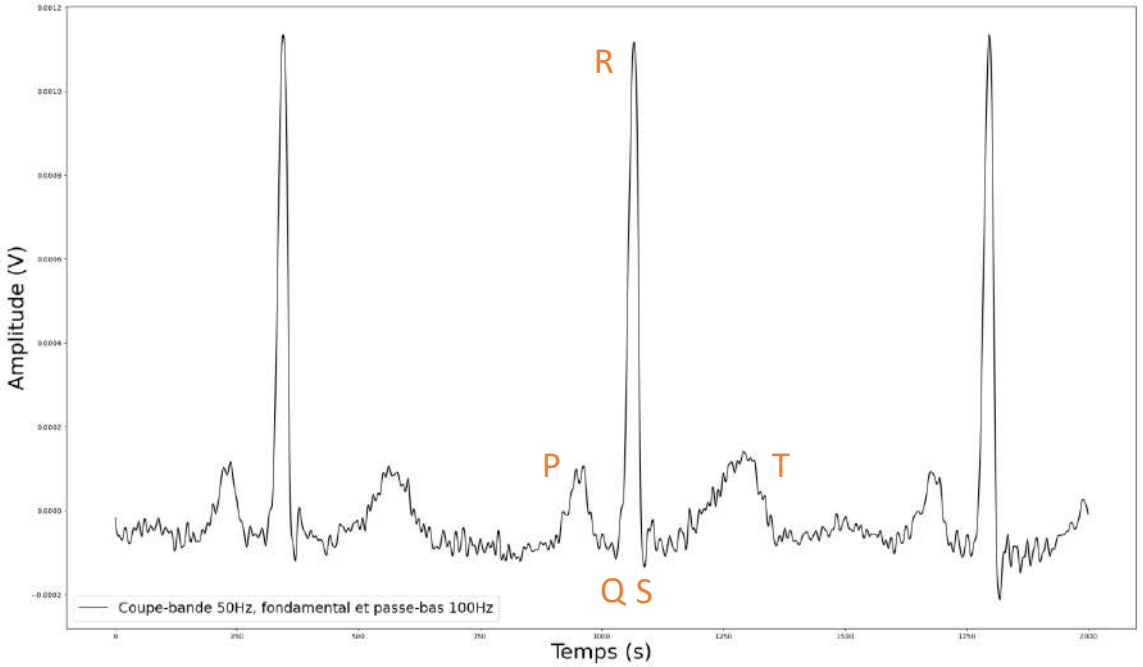
Filtres



+ Filtre passe-bas 100 Hz



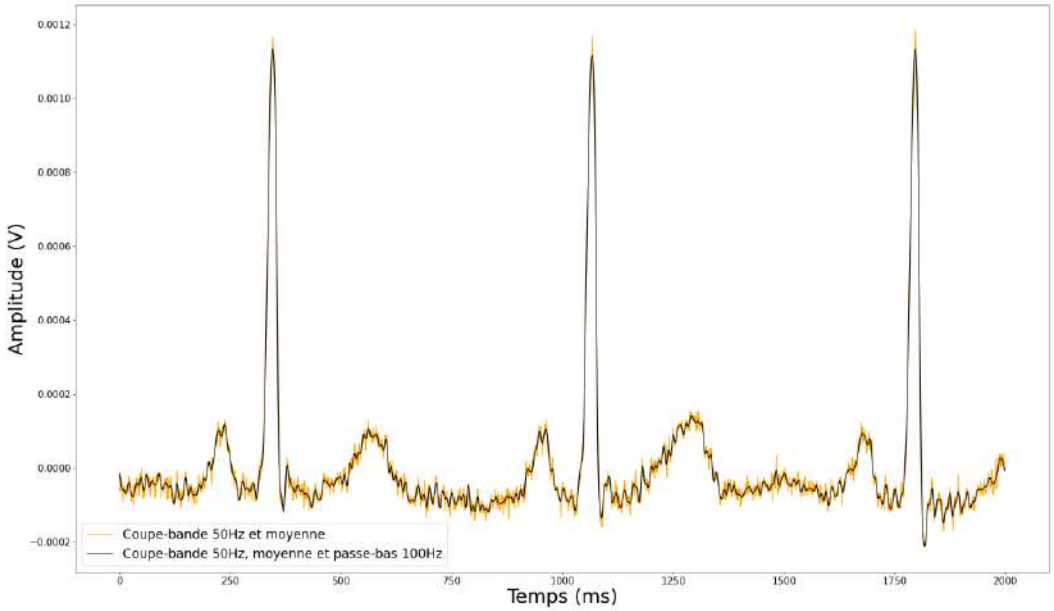




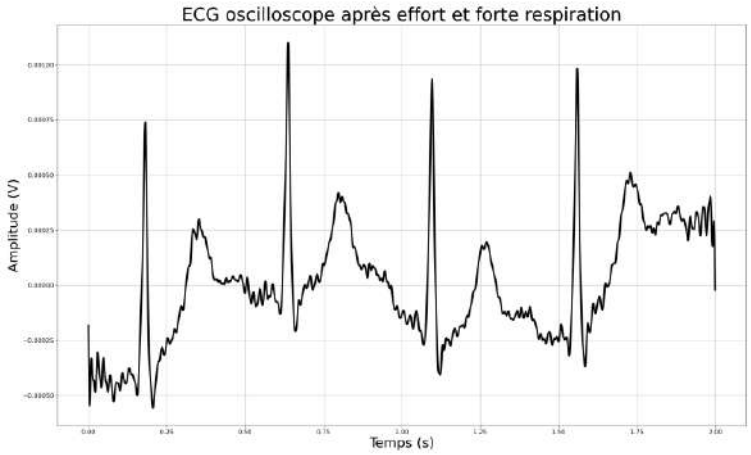
Filtres

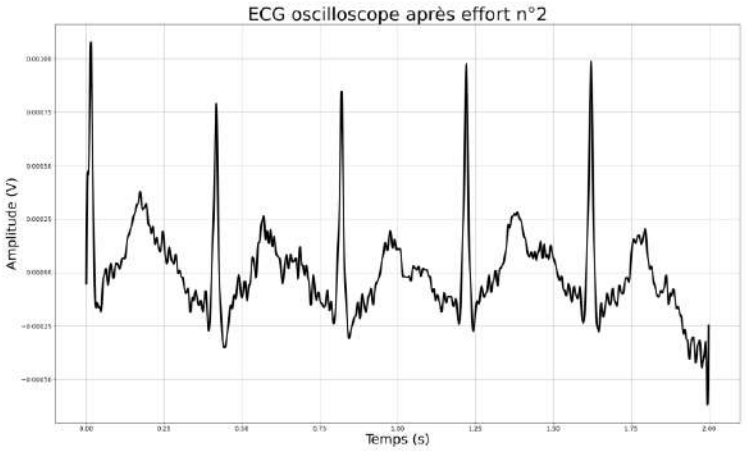
Comparaison :

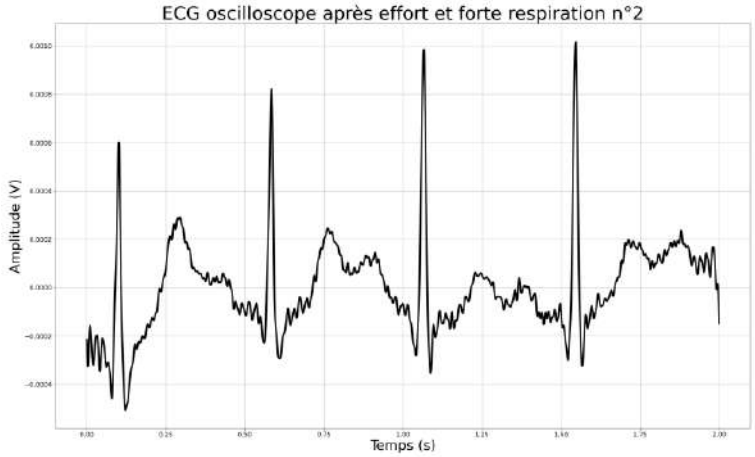
- Filtre coupe-bande 50 Hz et moyenne
- Filtre coupe-bande 50 Hz et moyenne, et passe bas 100Hz

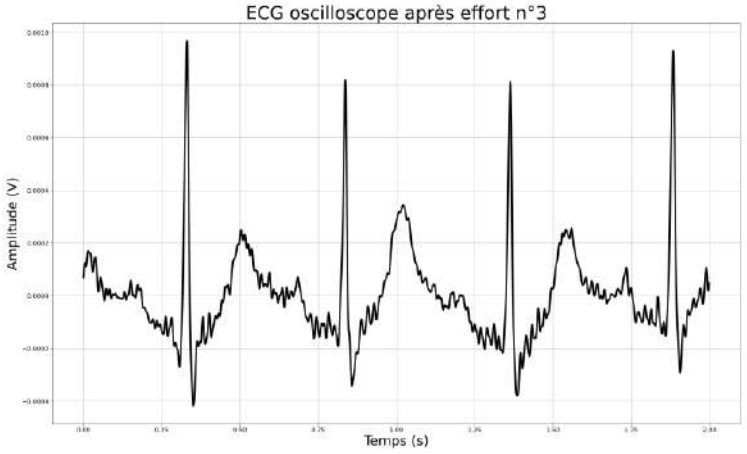


Application des filtres sur différentes courbes récupérées





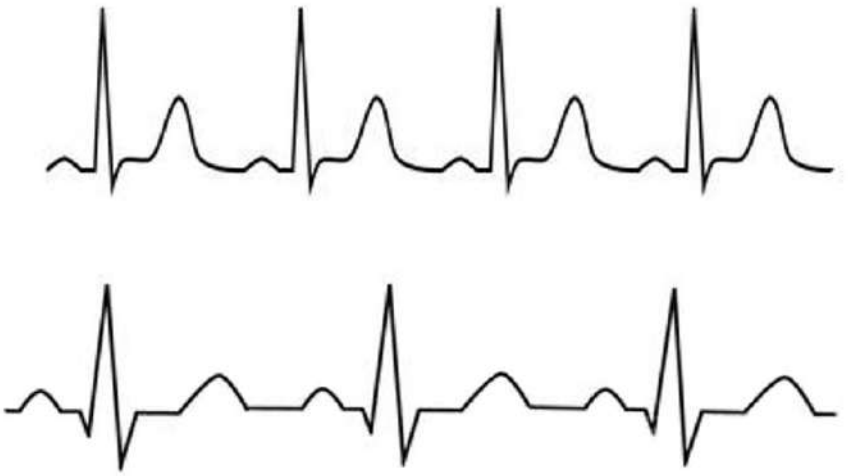




IV. Analyse et traitement du signal

- Caractéristiques à détecter
- Exemples d'ECG typiques :
  - ECG au repos
  - ECG après effort
  - ECG fibrillation ventriculaire
  - ECG tachycardie ventriculaire
  - ECG asystolie

Caractéristiques à détecter - bpm



1. Tachycardie : à partir de 180 bpm au repos
2. Rythme normal

```
def périodebpm(nom):
```

```
        for k in range (1,longlpics):
            periodes+=lpics[k][0]-lpics[k-1][0]
            # On détermine le temps entre chaque pic et on additionne toutes
            # les valeurs ensemble
        T=periodes/(longlpics-1)
        # On divise par le nombre d'intervalles pour faire la moyenne
        # C'est notre période
        bpm=60//T
        # On divise 1 minute par notre période pour obtenir le nombre de bpm
```

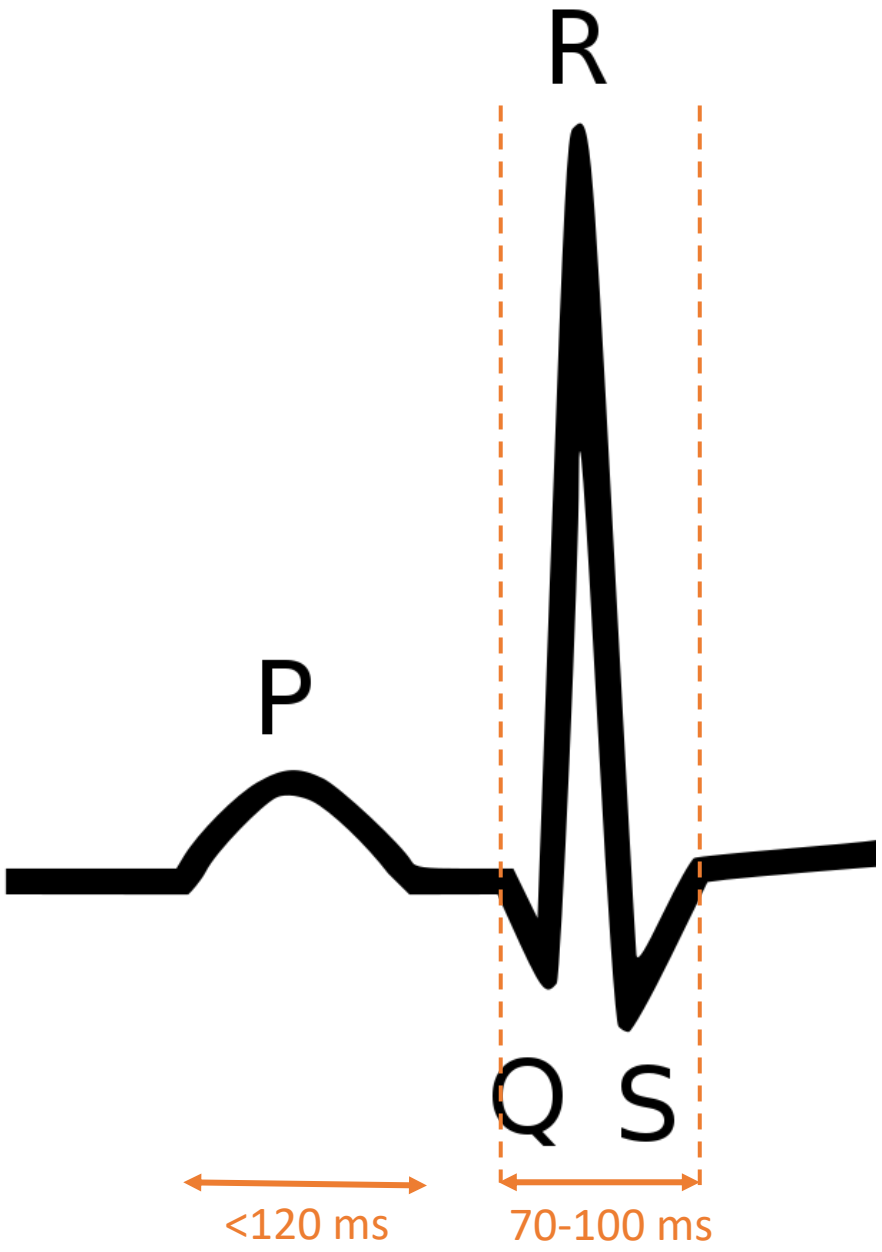
Caractéristiques à détecter - QRS

```
def tailleQRS(nom):
```

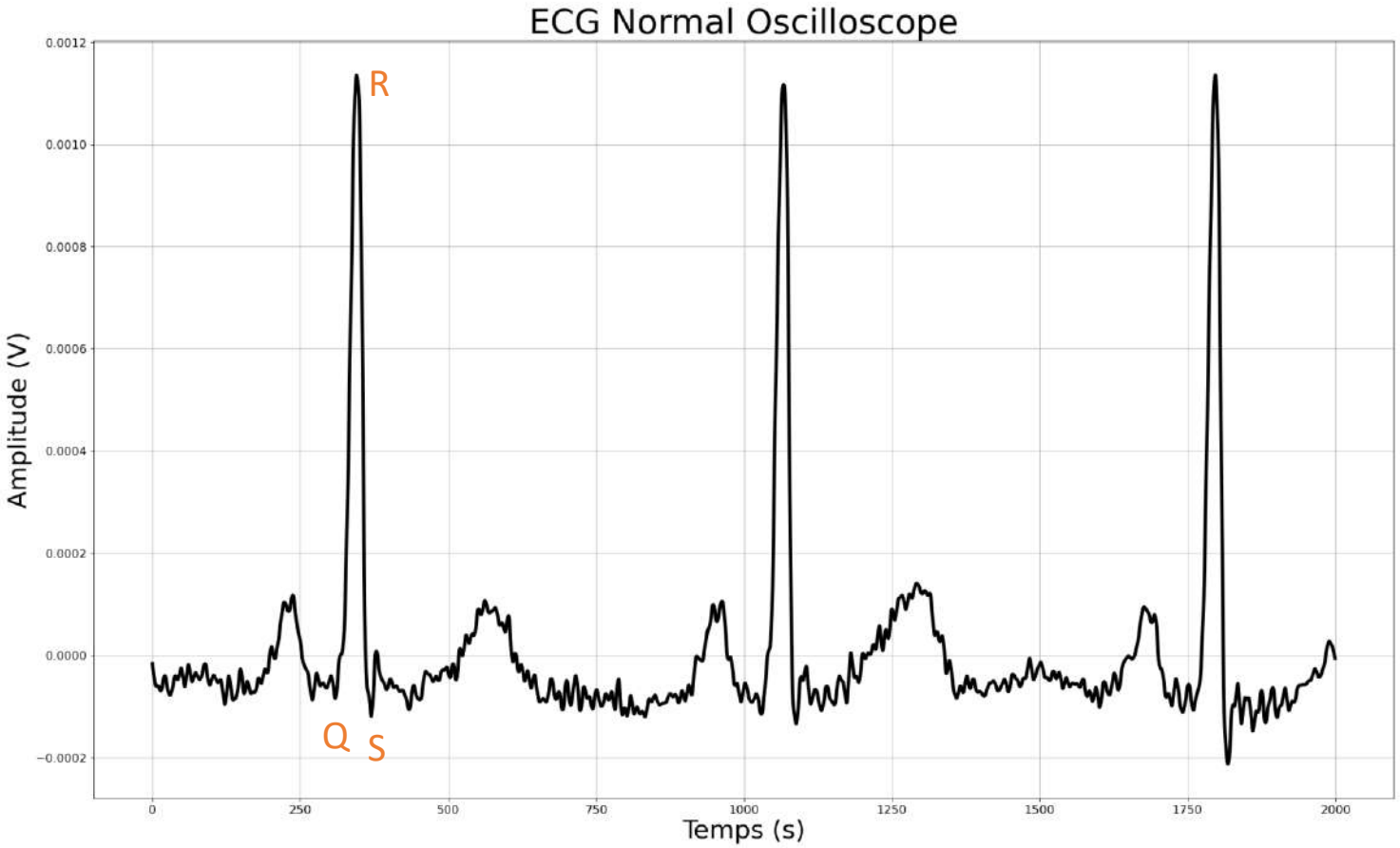
```
146                 a=[lignes[k][0],lignes[k][1]]
147                 lQ.append(a)
```

```
177                 b=[lignes[k][0],lignes[k][1]]
178                 lS.append(b)
```

```
295     for i in range (len(lS)):
296         lQRS.append(lS[i][0]-lQ[i][0])
297     if lQRS ==[]:
298         return 0
299     else:
300         QRS=sum(lQRS)/len(lQRS)
301         return QRS
302
```



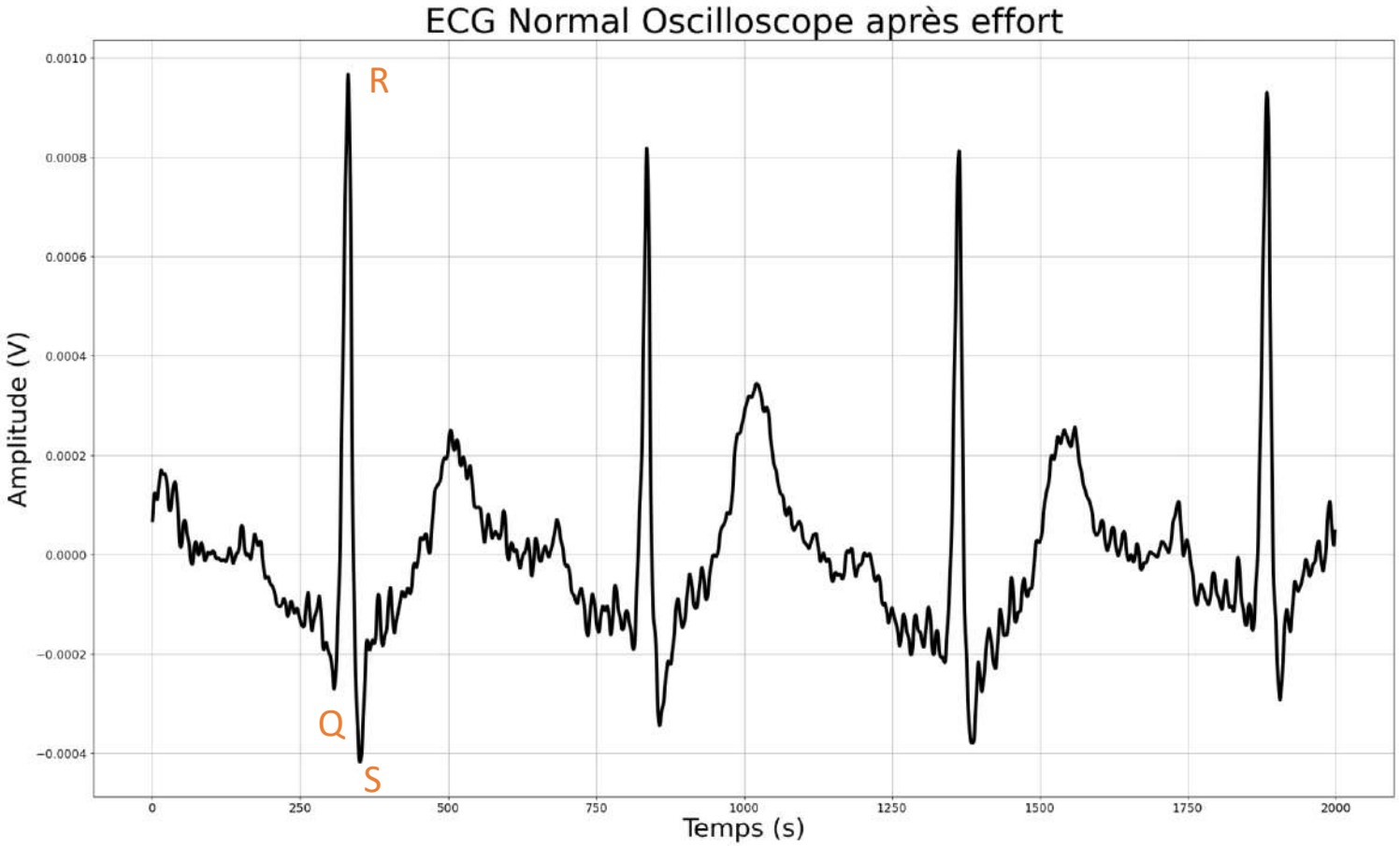
ECG normal au repos



BPM : 82
QRS : 77 ms

NON CHOQUABLE

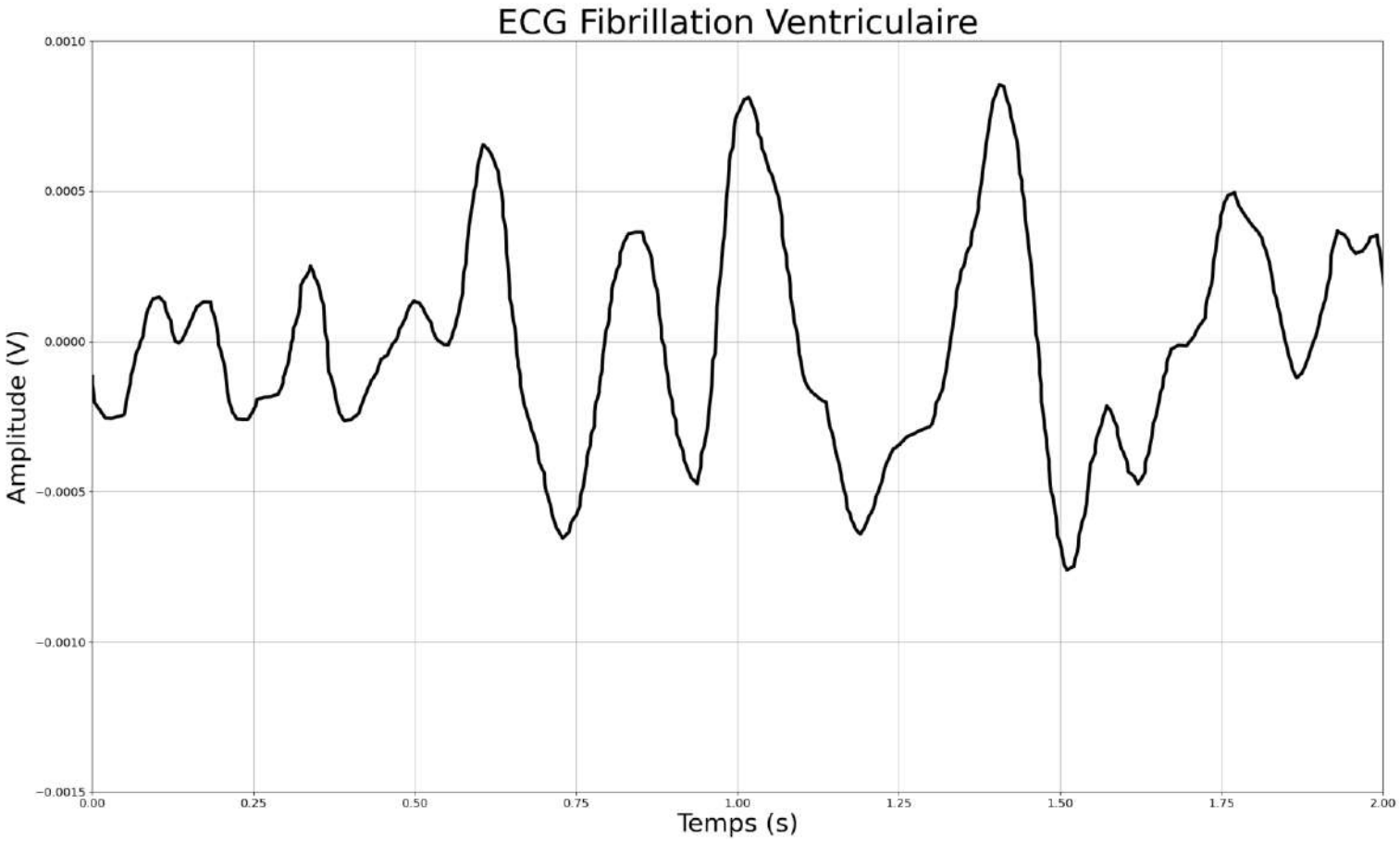
ECG normal après effort



BPM : 115
QRS : 72 ms

NON CHOQUABLE

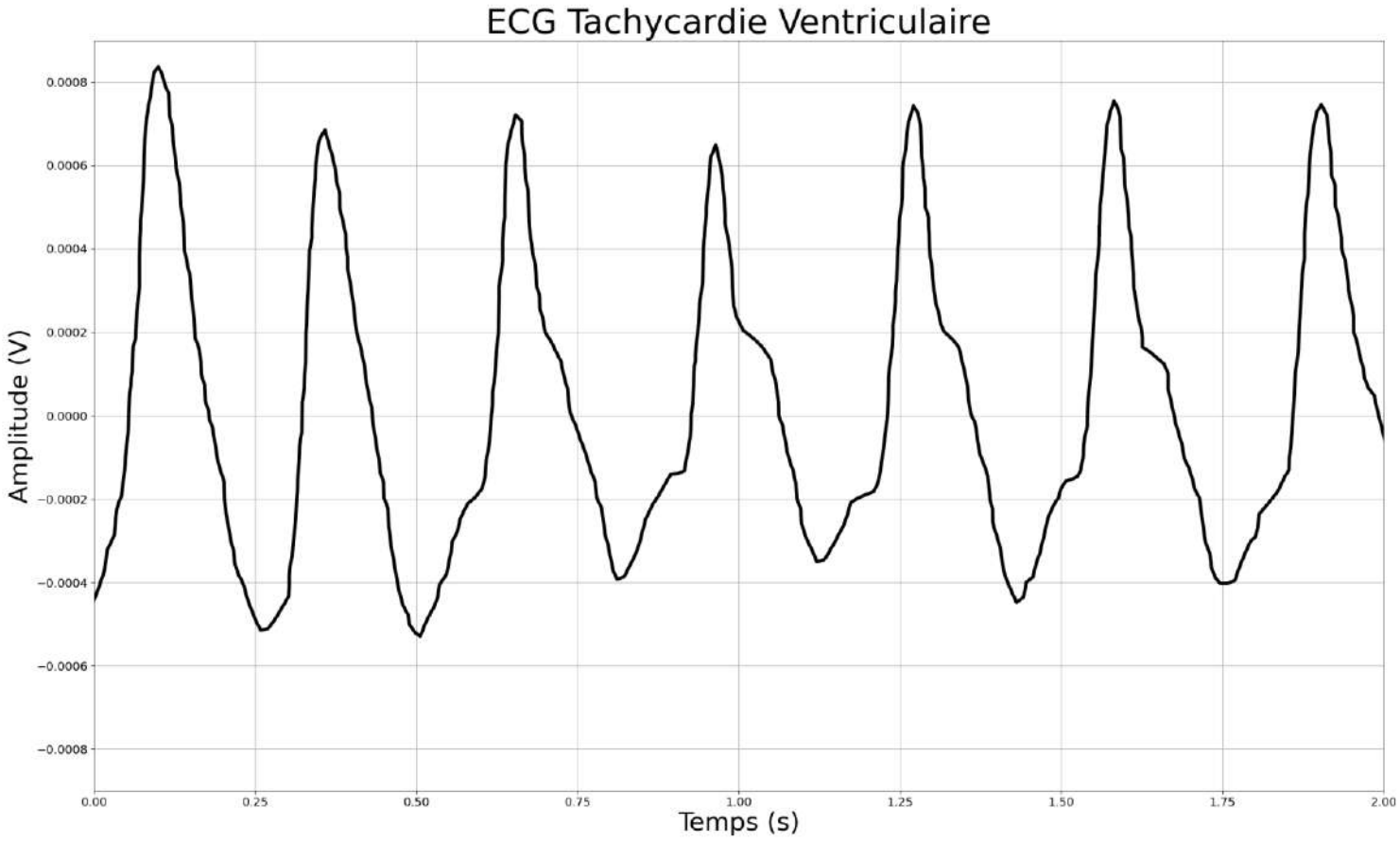
Fibrillation ventriculaire



BPM: 300
QRS : 570 ms

CHOQUABLE

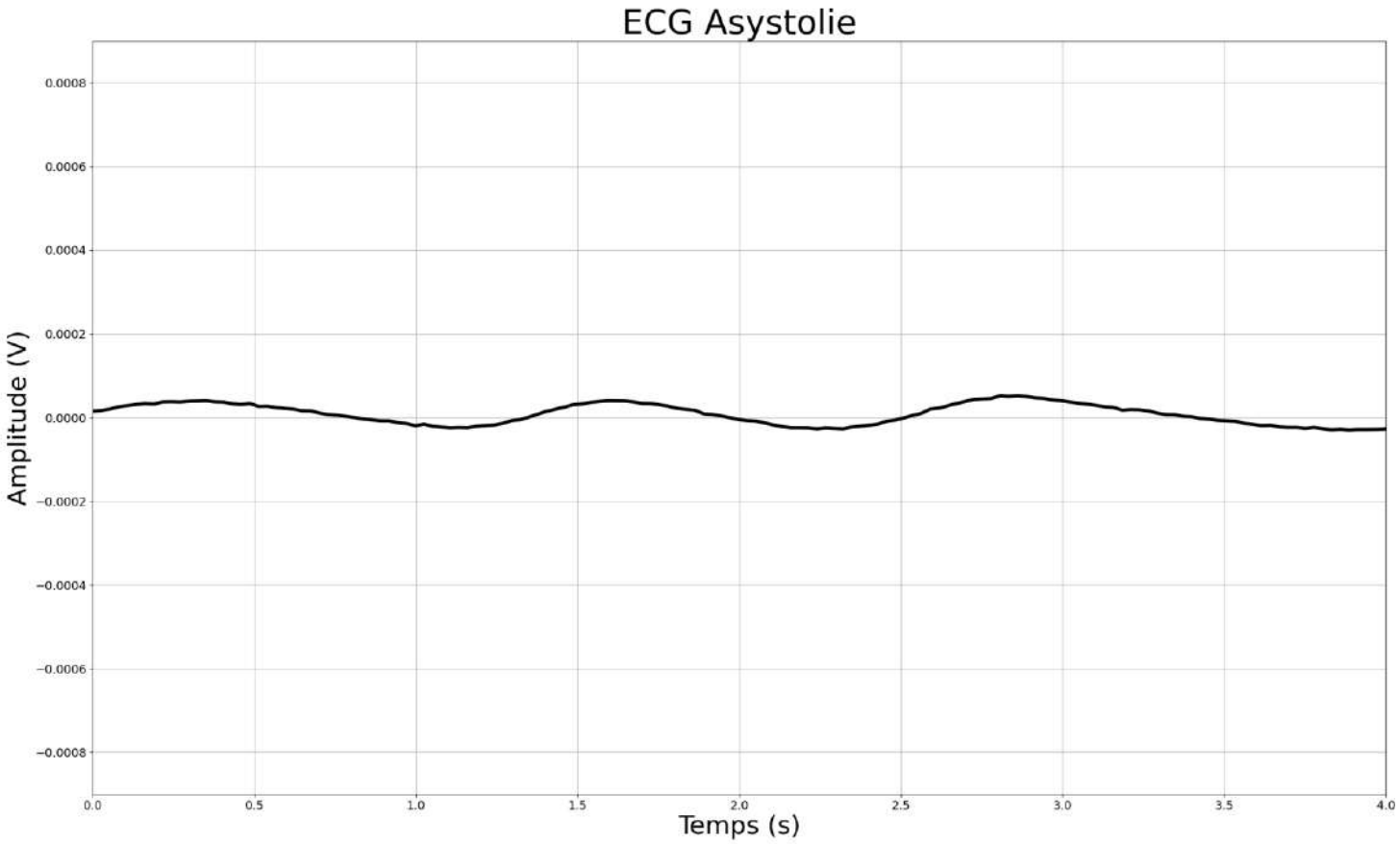
Tachycardie ventriculaire



BPM: 288
QRS : 690 ms

CHOQUABLE

Asystolie



BPM: 0
QRS : 0

NON CHOQUABLE

Bilan récapitulatif des différents cas d'ECG possibles et leur sortie

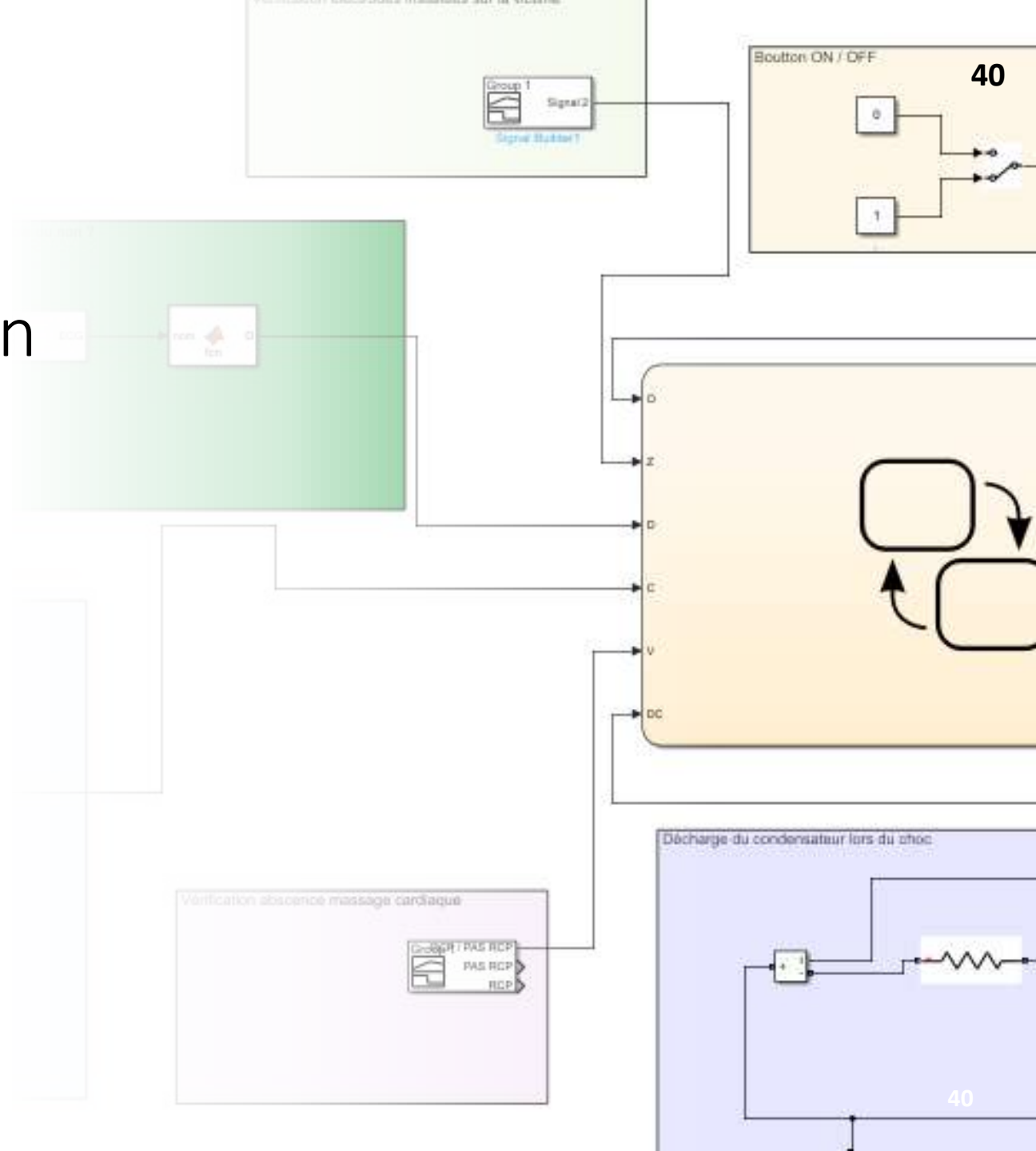
NOM	BPM	QRS	SORTIE DU DAE
ECG normal	< 180 bpm	> 50 ms	NON CHOQUABLE
Fibrillation ventriculaire	> 180 bpm	None	CHOQUABLE
Tachycardie ventriculaire	> 180 bpm	None	CHOQUABLE
Asystolie	0	None	NON CHOQUABLE

V. Modélisation

Matlab

+

Python

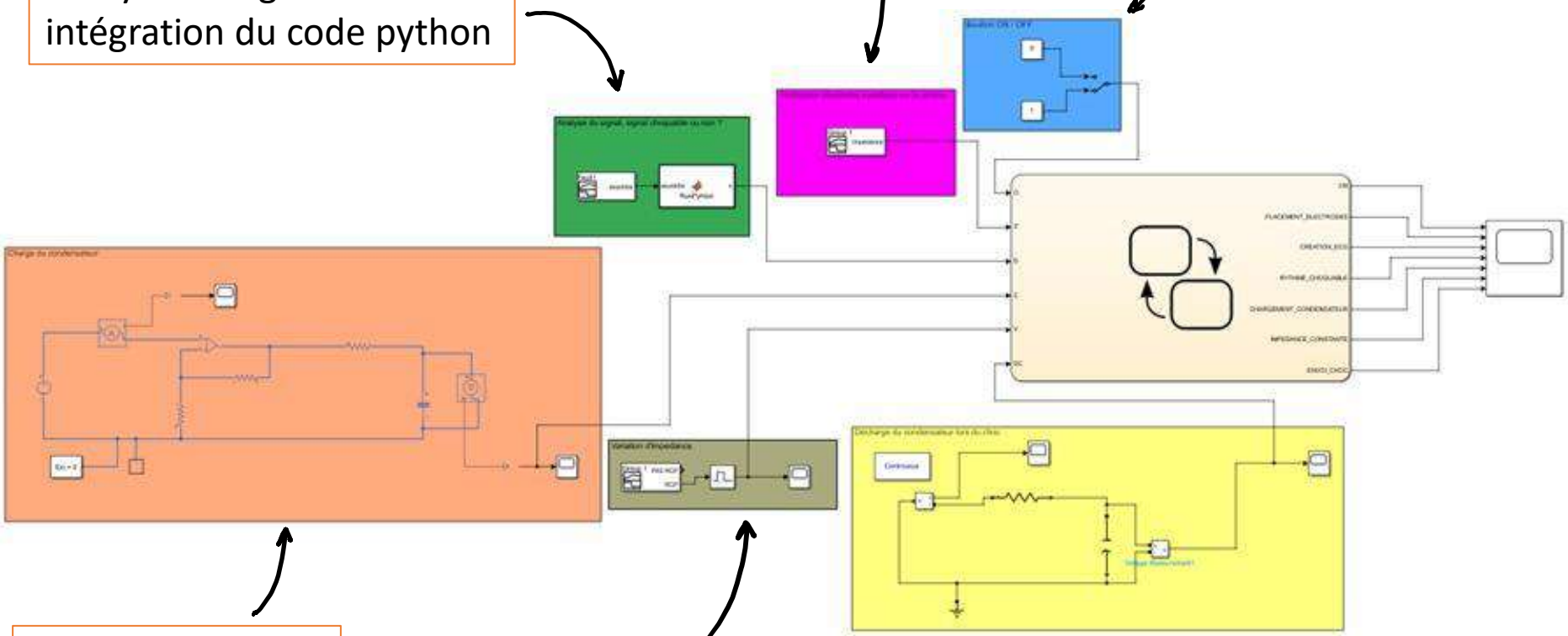


Modélisation

Vérification Impédance

Bouton On/Off

Analyse du signal : intégration du code python

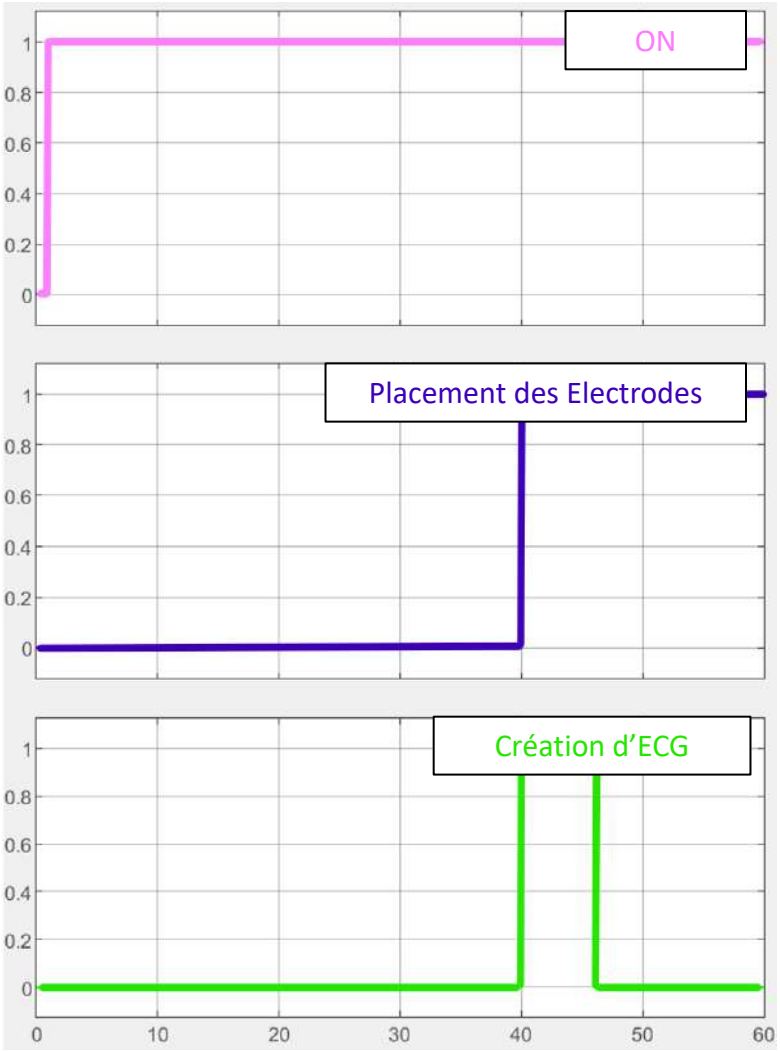


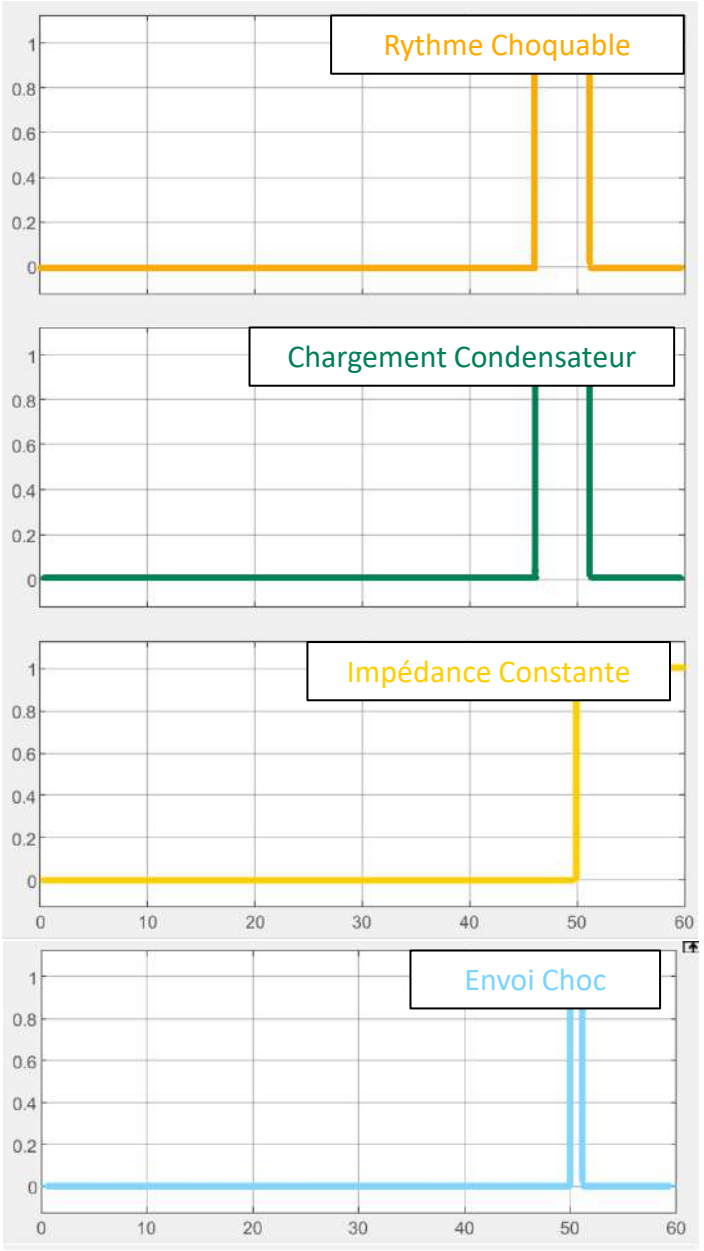
Charge du condensateur

Vérification variation d'impédance

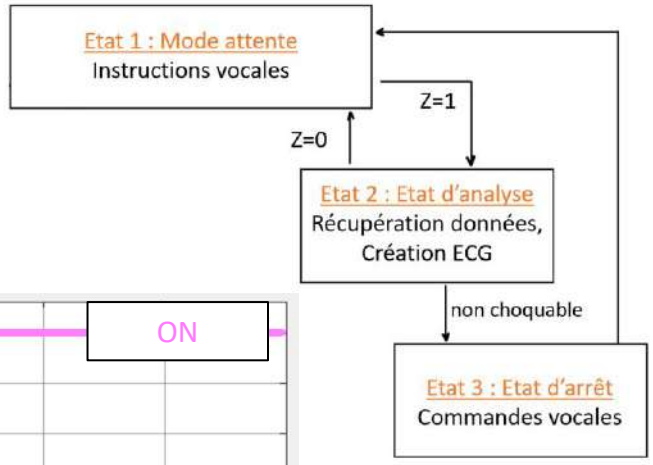
Décharge du condensateur

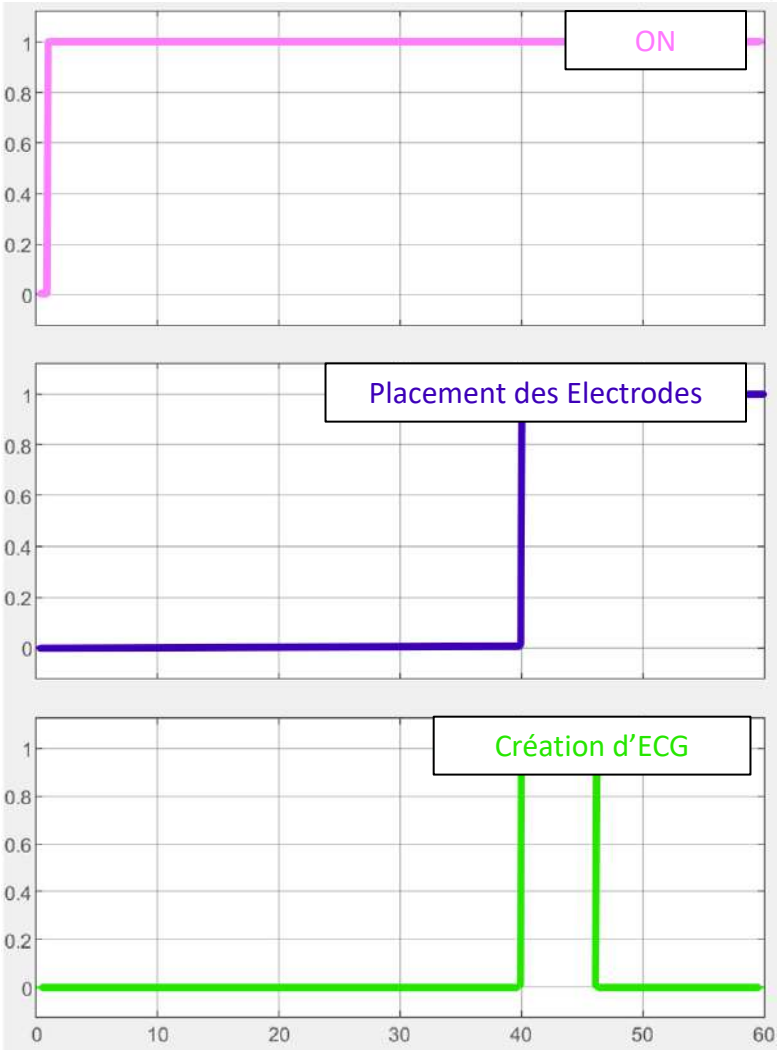
Rythme choquable, pas de RCP en cours : Déroulement normal

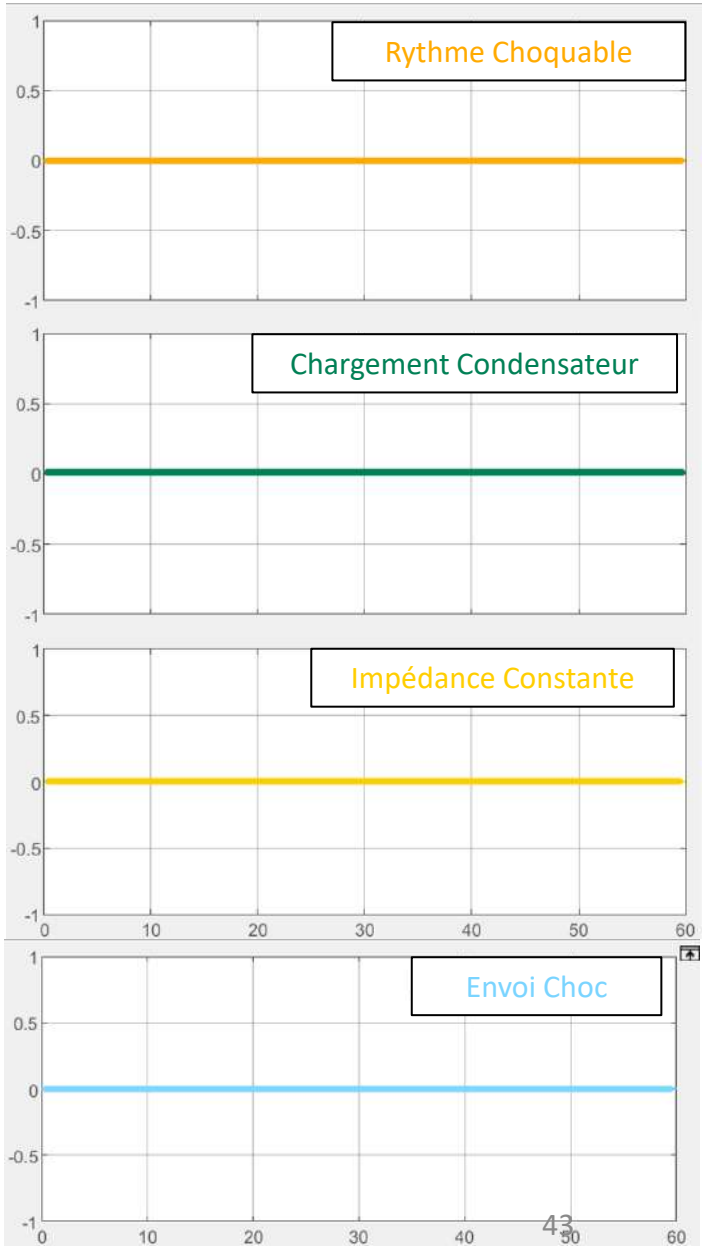




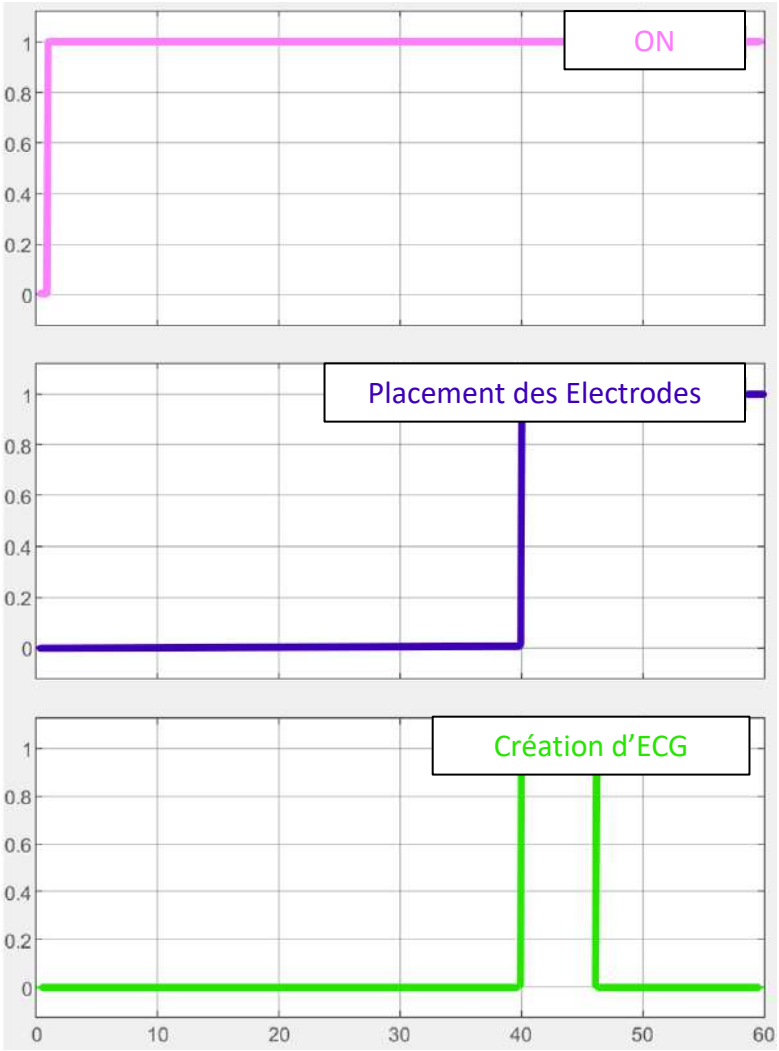
Rythme non choquable

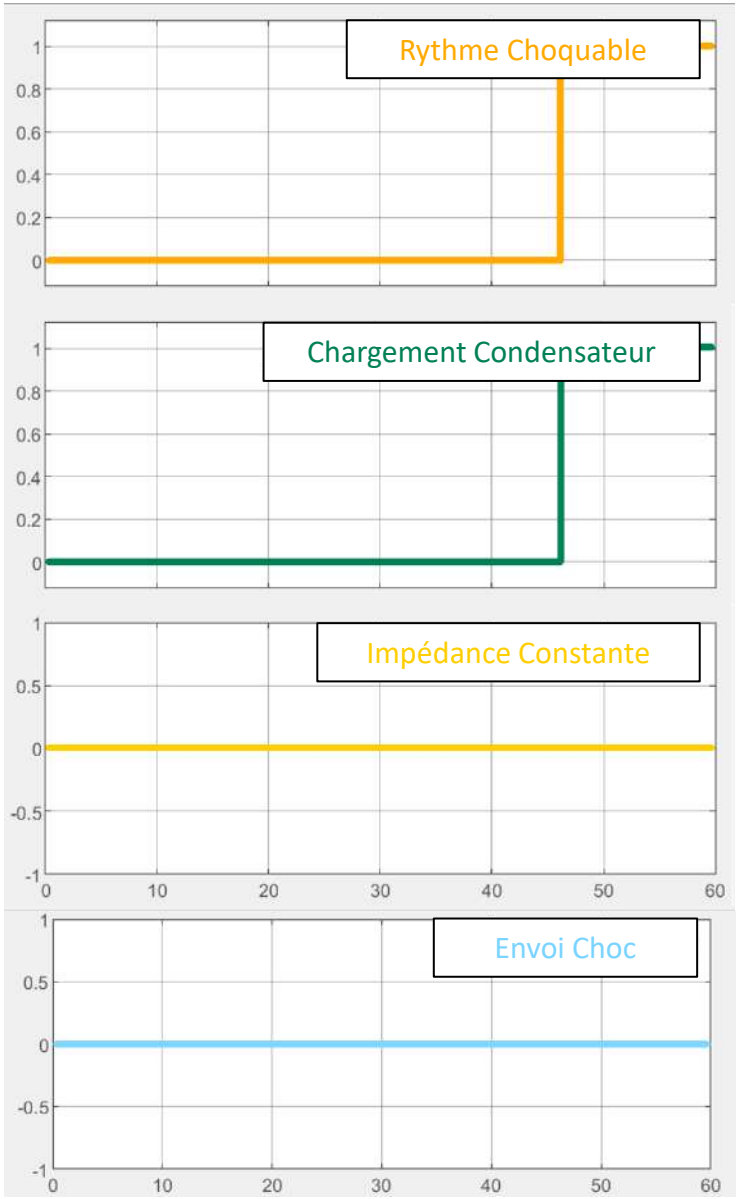






Rythme choquable, RCP en cours





Etat 4 : Etat de charge
Charge du condensateur

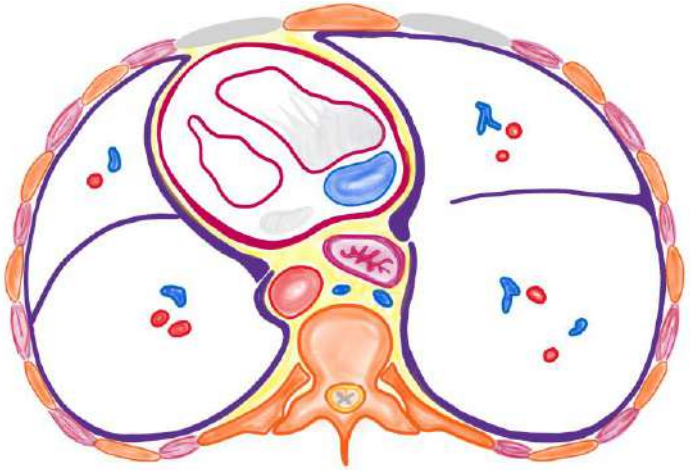
Zvar=0

Etat 5 : Etat de choc
Envoie du choc

Perspectives

Mesure de l'impédance thoracique :

- Adaptation de l'énergie délivrée en fonction de la masse corporelle
- Dépend de la conductivité des matériaux



Coupe transversale au niveau de T8

Tissus	Conductivité σ	Permittivité ϵ_r
Sang	0.7	$5{,}3 * 10^3$
Os compact	$2 * 10^{-2}$	$2{,}7 * 10^3$
Os spongieux	$8{,}2 * 10^{-2}$	$1{,}2 * 10^4$
Graisse mammaire	$2{,}4 * 10^{-2}$	$1{,}1 * 10^4$
Graisse	$2{,}2 * 10^{-2}$	$2{,}4 * 10^4$
Cœur	0,11	$3{,}5 * 10^5$
Poumon gonflé	0,22	$2{,}5 * 10^5$
Poumon dégonflé	$8 * 10^{-2}$	$1{,}4 * 10^5$
Muscles	0,32	$4{,}3 * 10^5$
Œsophage	0,52	$5{,}6 * 10^4$

Pour $1\ kHz < f < 100\ kHz$

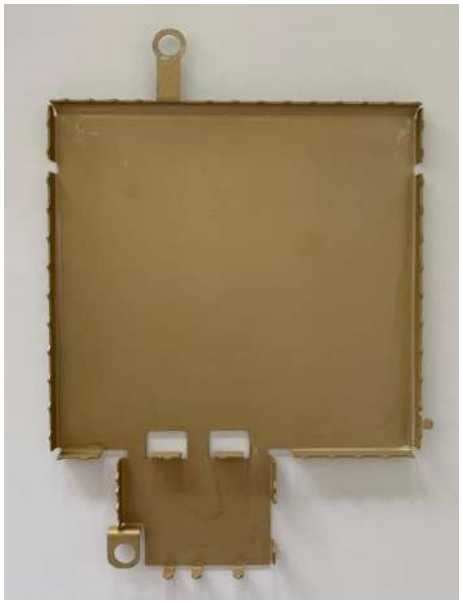
Annexes

Arrêt cardiaque sont suite à :

- Tachycardie
- Dysfonctionnement des ventricules
- Infarctus : caillot de sang bloquant l'irrigation
- Maladie rare (type Brugada : troubles électriques du rythme cardiaque)
- Hémorragie
- Intoxication ou noyade

Isolant Galvanique

Aucune liaison conductrice entre les 2 circuits



Placement des électrodes

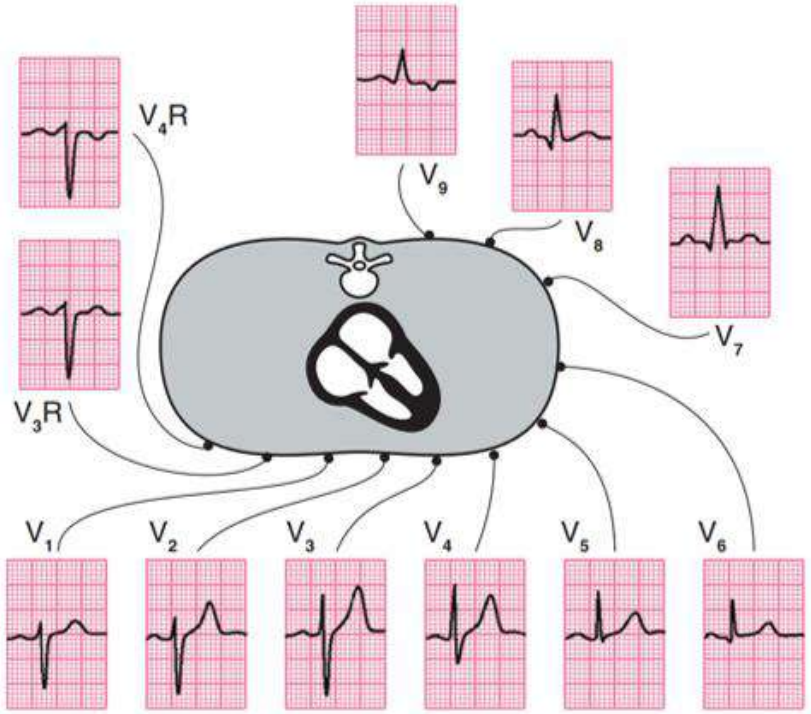
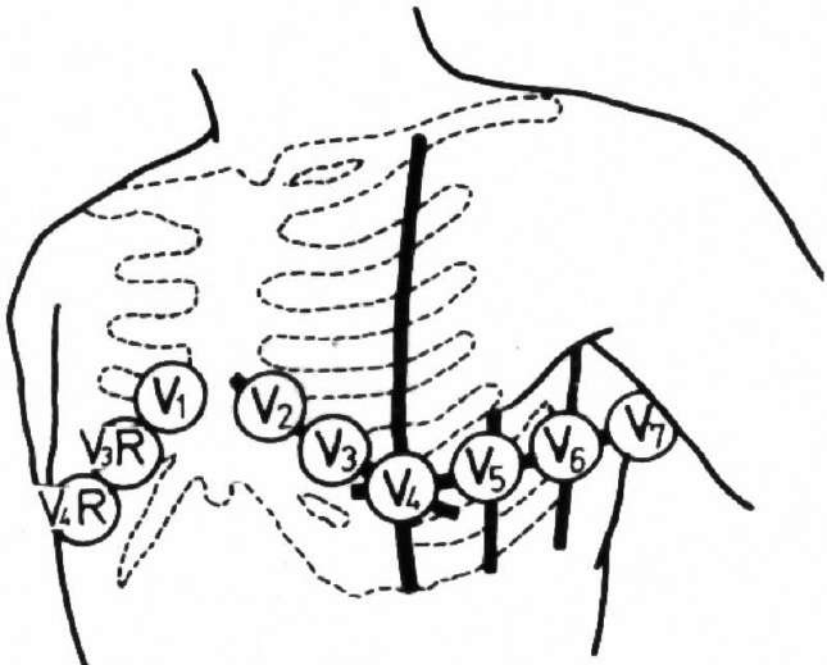
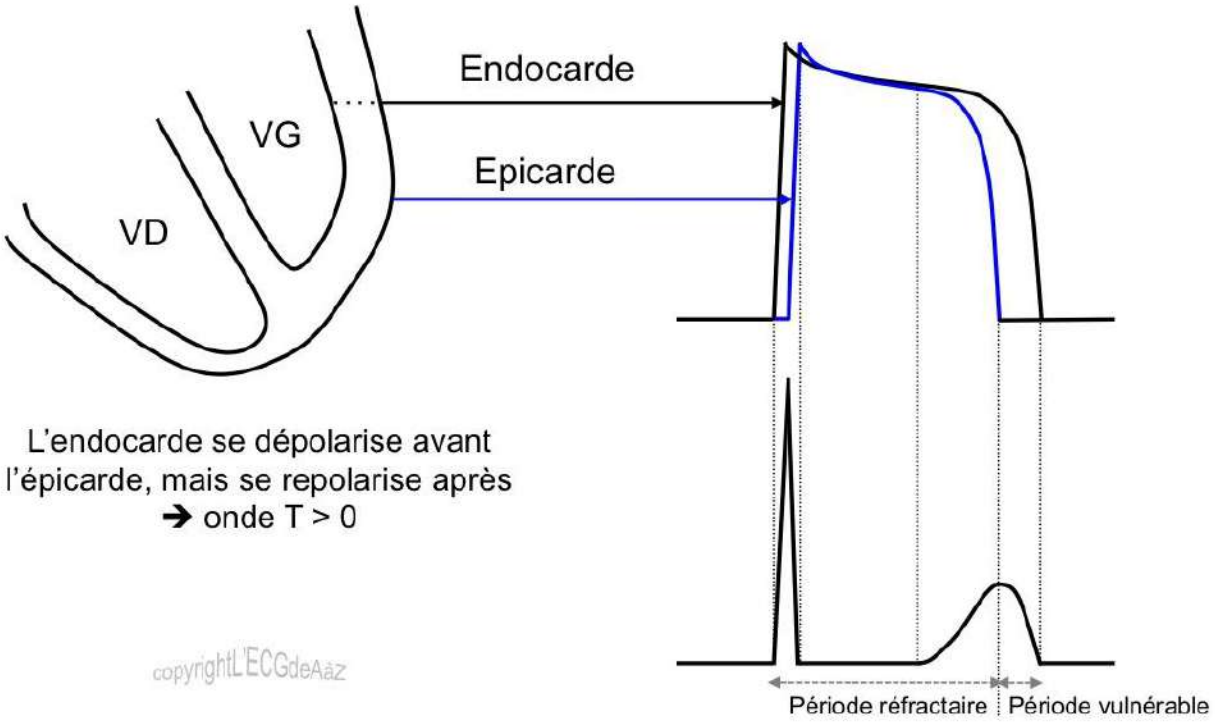


Figure 1.4. ECG 18 dérivations.

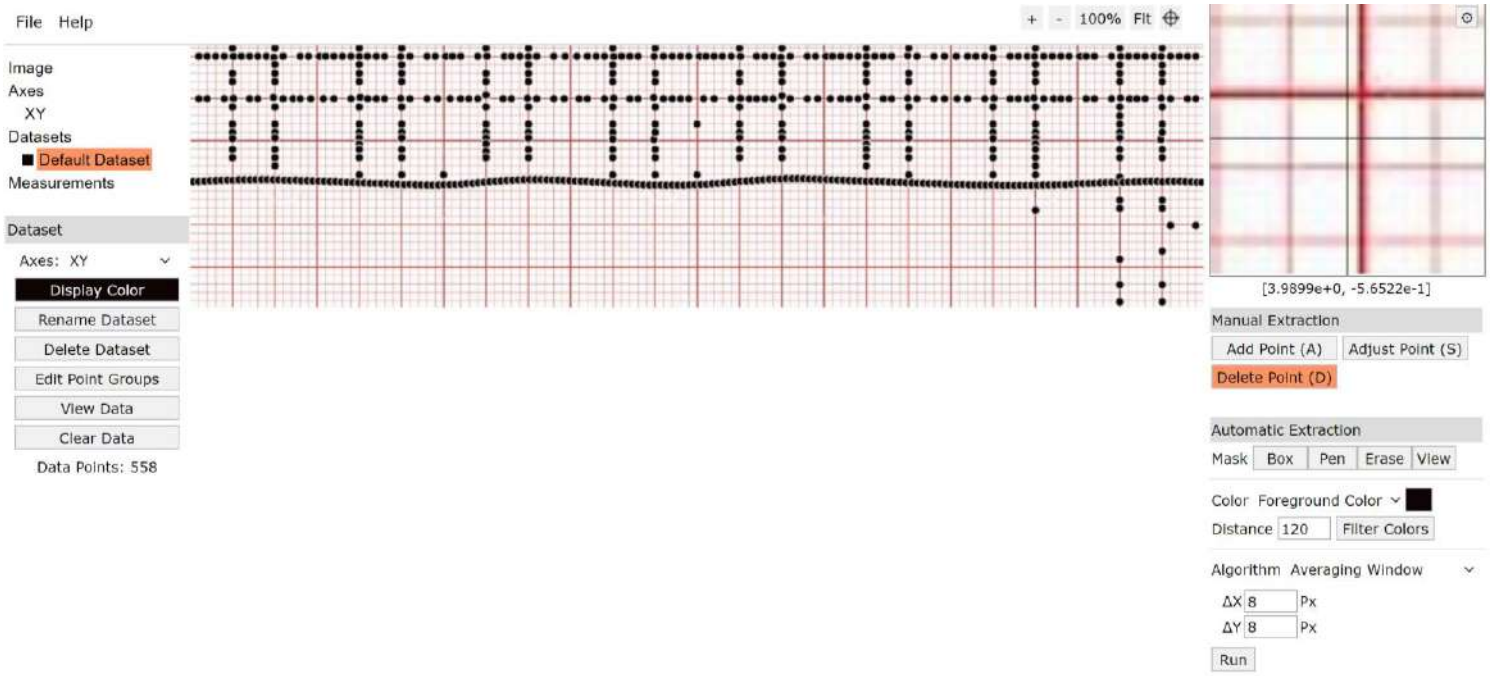


Dépolarisation et repolarisation



Récupération de signaux pré-existant

https://automeris.io/WebPlotDigitizer/



Modif valeurs courbes

```
 8  def modiftachy():
 9      f=open('tachycardie-ventriculaire - Copie2.txt',"w")
10      with open('tachycardie-ventriculaire - Copie.txt','r') as r:
11          lignes = r.readlines()
12      lig = []
13      for k in lignes:
14          s = ''
15          for c in k:
16              if c!=' ':
17                  s = s+c
18                  # On met dans s à la suite les uns les autres, tous les
19                  # caractères sauf les espaces
20          s = s.replace(',','.')
21          s = s.replace(';','\n')
22          lst = s.split('\n')
23          # On sépare tous les termes au niveau des \n (changement de ligne)
24          L = []
25          for el in lst:
26              if el!='' and el!='\n':
27                  L.append(float(el))
28          lig.append(L)
29      lignes = []
30      for L in lig:
31          if len(L)!=0:
32              lignes.append(L)
33      for i in range (len(lignes)):
34          lignes[i][0]=lignes[i][0]+0.390951061865189 24
35          lignes[i][1]=lignes[i][1]*10**(-3)
36          f.write(str(lignes[i][0])+";"+str(lignes[i][1])+'\n')
37
```

```
38  def modifasyst():
39      f=open('asystolie - nv.txt',"w")
40      with open('asystolie.txt','r') as r:
41          lignes = r.readlines()
42      lig = []
43      for k in lignes:
44          s = ''
45          for c in k:
46              if c!=' ':
47                  s = s+c
48                  # On met dans s à la suite les uns les autres, tous les
49                  # caractères sauf les espaces
50          s = s.replace(',','.')
51          s = s.replace(';','\n')
52          lst = s.split('\n')
53          # On sépare tous les termes au niveau des \n (changement de ligne)
54          L = []
55          for el in lst:
56              if el!='' and el!='\n':
57                  L.append(float(el))
58          lig.append(L)
59      lignes = []
60      for L in lig:
61          if len(L)!=0:
62              lignes.append(L)
63      for i in range (len(lignes)):
64          lignes[i][0]=lignes[i][0]+0.3857308677845832
65          lignes[i][1]=lignes[i][1]*10**(-3)
66          f.write(str(lignes[i][0])+";"+str(lignes[i][1])+'\n')
67
```

```
def modiffibril():
    f=open('Fibrillation-ventriculaire - nv.txt',"w")
    with open('Fibrillation-ventriculaire.txt','r') as r:
        lignes = r.readlines()
    lig = []
    for k in lignes:
        s = ''
        for c in k:
            if c!=' ':
                s = s+c
                # On met dans s à la suite les uns les autres, tous les
                # caractères sauf les espaces
        s = s.replace(',','.')
        s = s.replace(';','\n')
        lst = s.split('\n')
        # On sépare tous les termes au niveau des \n (changement de ligne)
        L = []
        for el in lst:
            if el!='' and el!='\n':
                L.append(float(el))
        lig.append(L)
    lignes = []
    for L in lig:
        if len(L)!=0:
            lignes.append(L)
    for i in range (len(lignes)):
        lignes[i][0]=lignes[i][0]+0.159091910954375l4
        lignes[i][1]=lignes[i][1]*10**(-3)
        f.write(str(lignes[i][0])+";"+str(lignes[i][1])+'\n')
```

Filtres numériques

```
8  ################### TRACE OSCILLOSCOPE SANS FILTRE ############################
9  import matplotlib.pyplot as plt
10 import numpy as np
11 with open('axeducoeur.txt','r') as f:
12     lignes = f.readlines()
13 #print(lignes)
14
15 list1, list2 ,liste, pourfourier = [], [], [],[]
16 for k in range (len(lignes)):
17     lst = lignes[k].split(',')
18     list1.append(float(lst[0])+1)
19     # Créé une liste de valeurs temporelles
20     list2.append(float(lst[1]))
21     # Créé une liste de valeurs d'amplitudes
22     liste.append([list1[k],list2[k]])
23     # Crée des listes de listes associant le temps avec l'amplitude
24 n=len(liste)
25
26 #Tracer l'ECG
27 plt.figure()
28 plt.title ('Electrocardiogramme avec bruit',fontsize=40)
29 plt.xlabel('Temps (s)',fontsize=30)
30 plt.ylabel('Amplitude (V)',fontsize=30)
31 plt.plot (list1,list2, color='black',linewidth=4)
32
```

```
34  ################### TRACE OSCILLOSCOPE AVEC FILTRE ##############################
35  def filtre(numero):
36      fourier= np.fft.fft(np.array(list2))
37      # Création de la transformé de fourier du signal
38      if numero ==1:
39          filtre=np.concatenate((np.ones(94),np.zeros(1812),np.ones(94)))
40          # Création du filtre
41          transffourier= np.multiply(fourier,filtre)
42          # Application des filtres sur le signal spectral
43          signal = np.fft.ifft(transffourier)
44          # Transformation du signal spectral en signal temporel
45          nom="Passe-bas 50 Hz"
46          colo='red'
47      elif numero ==2:
48          filtre=np.concatenate((np.zeros(2),np.ones(94),np.zeros(1808),np.ones(94),np.zeros(2)))
49          transffourier= np.multiply(fourier,filtre)
50          signal = np.fft.ifft(transffourier)
51          nom="Passe-bas 50Hz et coupe bande moyenne"
52          colo='green'
53      elif numero ==3:
54          filtre=np.concatenate((np.ones(94),np.zeros(10),np.ones(1792),np.zeros(10),np.ones(94)))
55          transffourier= np.multiply(fourier,filtre)
56          signal = np.fft.ifft(transffourier)
57          nom="Coupe-bande 50Hz"
58          colo='dodgerblue'
59      elif numero ==4:
60          filtre=np.concatenate((np.zeros(2),np.ones(93),np.zeros(10),np.ones(1791),np.zeros(10),
61                                 np.ones(93),np.zeros(1)))
62          transffourier= np.multiply(fourier,filtre)
63          signal = np.fft.ifft(transffourier)
64          nom="Coupe-bande 50Hz et moyenne"
65          colo='orange'
66      elif numero ==5:
67          filtre=np.concatenate((np.zeros(2),np.ones(93),np.zeros(10),np.ones(105),np.zeros(1581),
68                                 np.ones(105),np.zeros(10),np.ones(93),np.zeros(1)))
69          transffourier= np.multiply(fourier,filtre)
70          signal = np.fft.ifft(transffourier)
71          nom="Coupe-bande 50Hz, moyenne et passe-bas 100Hz"
72          colo='black'
```

```
    plt.figure()
    plt.plot(fourier,color='orange',linewidth=4)
    plt.plot(filtre,color='mediumorchid', linewidth=4)
    plt.plot(transffourier,color='navy',linewidth=3)
    plt.xlabel('Points',fontsize=30)
    plt.ylabel('Amplitude',fontsize=30)
    for tickLabel in plt.gca().xaxis.get_ticklabels():
        tickLabel.set_fontsize(16)
    for tickLabel in plt.gca().yaxis.get_ticklabels():
        tickLabel.set_fontsize(16)
    plt.title("Signal Spectral", fontsize=40)
    plt.legend(fontsize=20)

    plt.figure()
    plt.plot(np.real(signal), label=nom, color=colo,  linewidth=1)
    plt.xlabel('Temps (ms)',fontsize=30)
    plt.ylabel('Amplitude (V)',fontsize=30)
    for tickLabel in plt.gca().xaxis.get_ticklabels():
        tickLabel.set_fontsize(15)
    for tickLabel in plt.gca().yaxis.get_ticklabels():
        tickLabel.set_fontsize(15)
    plt.title("Signal Temporel Filtré", fontsize=40)
    plt.legend(fontsize=20)

```

Tracés courbes

```
4  ################### ECG NORMAL ##################
5
6  def ECGnorm(txt):
7      with open(txt,'r') as f:
8          lignes = f.readlines()
9          f.close()
10     lig = []
11     for k in lignes:
12         s = ''
13         for c in k:
14             if c!='\x00' and c!=chr(255) and c!=chr(254) and c!=chr(32):
15                 s = s+c
16         s = s.replace(',','.')
17         # On remplace les virgules par des points pour s'adapter à l'écriture
18         # de python
19         lst = s.split('\t')
20         L = []
21         for el in lst:
22             if el!='' and el!='\n':
23                 L.append(float(el))
24         lig.append(L)
25     lignes = []
26     for L in lig:
27         if len(L)!=0:
28             lignes.append(L)
29     return lignes
```

```
31
32  ################### ECG NORMAL PAR EXPERIMENTATION #################
33
34  def ECGoscillo(txt):
35      with open(txt,'r') as f:
36          ligne = f.readlines()
37          f.close()
38      list1, list2 ,liste = [], [], []
39      for k in range (len(ligne)):
40          lst = ligne[k].split(',')
41          list1.append(float(lst[0])+1)
42          # Créé une liste de valeurs temporelles
43          list2.append(float(lst[1]))
44          # Créé une liste de valeurs d'amplitudes
45          liste.append([list1[k],list2[k]])
46          # Crée des listes de listes associant le temps avec l'amplitude
47      plt.figure()
48      plt.grid()
49      fourier= np.fft.fft(np.array(list2))
50      # On créé la transformé de fourier de notre signal
51      filtre =np.concatenate((np.zeros(2),np.ones(93),np.zeros(10),np.ones(105),
52                              np.zeros(1581),np.ones(105),np.zeros(10),np.ones(93),np.zeros(1)))
53      # On créé un filtre symétrique
54      transffourier= np.multiply(fourier,filtre)
55      # Qu'on applique sur la transofrmée de Fourier
56      signal = np.fft.ifft(transffourier)
57      # On retransforme notre signal en amplitude en fonction du temps
58      if txt=='axeducoeur.txt':
59          titre='ECG oscilloscope au repos'
60      elif txt=='Amandine1.txt':
61          titre='ECG oscilloscope après effort'
62      elif txt=='Amandine2.txt':
63          titre='ECG oscilloscope après effort n°2'
64      elif txt=='Amandine4.txt':
65          titre='ECG oscilloscope après effort et forte respiration'
66      elif txt=='Amandine5.txt':
67          titre='ECG oscilloscope après effort et forte respiration n°2'
68      elif txt=='Amandine7.txt':
69          titre='ECG oscilloscope après effort n°3'
70      plt.title (titre,fontsize=40)
71      plt.xlabel('Temps (s)',fontsize=30)
72      plt.ylabel('Amplitude (V)',fontsize=30)
73      for tickLabel in plt.gca().xaxis.get_ticklabels():
74          tickLabel.set_fontsize(14)
75      for tickLabel in plt.gca().yaxis.get_ticklabels():
76          tickLabel.set_fontsize(14)
77      plt.plot(list1,np.real(signal),linewidth=4, color='black')
78
```

```
def ECG(txt):
    with open(txt,'r') as f:
        ligne = f.readlines()
        f.close()
    list1, list2 ,lignes = [], [], []
    for k in range (len(ligne)):
        lst = ligne[k].split(',')
        list1.append(float(lst[0]))
        # Créé une liste de valeurs temporelles
        list2.append(float(lst[1]))
        # Créé une liste de valeurs d'amplitudes
        lignes.append([list1[k],list2[k]])
        # Crée des listes de listes associant le temps avec l'amplitude
    return lignes

def trace(txt,P):
    if P==1 or P==2 or P==3:
        courbe=ECG(txt)
    else :
        courbe=ECGnorm(txt)
    n=len(courbe)
    tps=[courbe[i][0] for i in range (n)]
    Lpoints=[courbe[i][1] for i in range (n)]
    plt.figure()
    if P==0:
        plt.axis([0,10,-4e-4,6e-4])
        plt.title ('ECG Normal',fontsize=40)
    elif P==1:
        plt.axis([0,4,-0.9e-3,0.9e-3])
        plt.title ('ECG Asystolie',fontsize=40)
    elif P==2:
        plt.axis([0,2,-1.5e-3,1e-3])
        plt.title ('ECG Fibrillation Ventriculaire',fontsize=40)
    elif P==3:
        plt.axis([0,2,-0.9e-3,0.9e-3])
        plt.title ('ECG Tachycardie Ventriculaire',fontsize=40)
    for tickLabel in plt.gca().xaxis.get_ticklabels():
        tickLabel.set_fontsize(14)
    for tickLabel in plt.gca().yaxis.get_ticklabels():
        tickLabel.set_fontsize(14)
    plt.xlabel('Temps (s)',fontsize=30)
    plt.ylabel('Amplitude (V)',fontsize=30)
    plt.grid()
    plt.plot (tps,Lpoints, color='black',linewidth=4)
```

Filtres internes

```
 9  def ECG(nom):
10      if nom=='Asystolie':
11          txt='asystolie.txt'
12      elif nom=='Fibrillation ventriculaire':
13          txt='Fibrillation-ventriculaire.txt'
14      elif nom=='Tachycardie ventriculaire':
15          txt='tachycardie-ventriculaire.txt'
16      elif nom=='Oscillorepos':
17          txt='axeducoeur.txt'
18      elif nom=='Oscilloeffort1':
19          txt='Amandine1.txt'
20      elif nom=='Oscilloeffort2':
21          txt='Amandine2.txt'
22      elif nom=='Oscilloeffort3':
23          txt='Amandine4.txt'
24      elif nom=='Oscilloeffort4':
25          txt='Amandine5.txt'
26      elif nom=='Oscilloeffort5':
27          txt='Amandine7.txt'
28      if nom=='Asystolie' or nom=='Fibrillation ventriculaire' or nom=='Tachycardie ventriculaire':
29          with open(txt,'r') as f:
30              ligne = f.readlines()
31              f.close()
32          list1, list2 ,lignes = [], [], []
33          for k in range (len(ligne)):
34              lst = ligne[k].split(',')
35              list1.append(float(lst[0]))
36              # Créé une liste de valeurs temporelles
37              list2.append(float(lst[1]))
38              # Créé une liste de valeurs d'amplitudes
39              lignes.append([list1[k],list2[k]])
40              # Crée des listes de listes associant le temps avec l'amplitude
41          return lignes
42
```

```
    else:
        with open(txt,'r') as f:
            lignes = f.readlines()
        list1, list2 ,liste = [], [], []
        for k in range (len(lignes)):
            lst = lignes[k].split(',')
            list1.append(float(lst[0])+1)
            list2.append(float(lst[1]))
            liste.append([list1[k],list2[k]])
        fourier= np.fft.fft(np.array(list2))
        # On créé la transformé de fourier de notre signal
        filtre =np.concatenate((np.zeros(2),np.ones(93),np.zeros(10),np.ones(105),
                                np.zeros(1581),np.ones(105),np.zeros(10),np.ones(93),np.zeros(1)))
        # On créé un filtre symétrique
        transffourier= np.multiply(fourier,filtre)
        # Qu'on applique sur la transofrmée de Fourier
        signal = np.fft.ifft(transffourier)
        # On retransforme notre signal en amplitude en fonction du temps
        lignes=[]
        for i in range (len(list1)):
            lignes.append([list1[i],np.real(signal)[i]])
        return lignes
```

```
66  ################## ANALYSE ET DONNEES DES COURBES ##############
67
68  # Déterminerl'emplacement des pics et la période pour obtenir le nombre de
69  # battements par minute
70  def périodebpm(nom):
71      lignes=ECG(nom)
72      n1=len(lignes)
73      maxi=lignes[0][1]
74      lpics=[]
75      periodes=0
76      for i in range (n1):
77          if lignes[i][1]>6e-4:
78          #Lors d'un pic
79              if lignes[i][1]>maxi:
80              #On détermine le maximum
81                  maxi=lignes[i][1]
82              elif lignes[i][1]<maxi and lignes[i-1][1]==maxi:
83              #Lorsque le maximum est atteint, on met le temps et son maximum
84              # associé dans une liste lpics pour plus tard
85                  a=[lignes[i-1][0],i-1,lignes[i-1][1]]
86                  lpics.append(a)
87          else :
88          #Lorsqu'on est en dehors d'un pic, le repasse le maximum à 0
89          # pour ne pas comparer les maximums des pics entre eux
90              maxi=0
91      longlpics=len(lpics)
92      if lpics==[]:
93          return 0,[]
94      else :
95          for k in range (1,longlpics):
96              periodes+=lpics[k][0]-lpics[k-1][0]
97              # On détermine le temps entre chaque pic et on additionne toutes
98              # les valeurs ensemble
99          T=periodes/(longlpics-1)
100         # On divise par le nombre d'intervalles pour faire la moyenne
101         # C'est notre période
102         bpm=60//T
103         # On divise 1 minute par notre période pour obtenir le nombre de bpm
104         return bpm,lpics
```

```
200
201 # Déterminer la taille du complexe QRS
202 def tailleQRS(nom):
203     bpm,lpics=périodebpm(nom)
204     # On récupère les valeurs obtenues précédemment pour faciliter le calcul
205     lignes=ECG(nom)
206     lQ=[]
207     lS=[]
208     lQRS=[]
209     n=len(lpics)
210     for i in range (1,n-1):
211         mini1=lpics[i][2]
212         #Pour chaque pic, on définit un minimum qui se trouve au niveau du
213         # maximum actuel (càd au niveau d'un pic)
214         fin=0
215         k=lpics[i][1]-1
216         # On démarre au point juste avant le pic
217         while fin==0 and k!=0:
218             if mini1<=lignes[k][1] and mini1<=lignes[k-1][1]:
219                 # On cherche le moment où la courbe remonte, on met 2
220                 # conditions pour si les courbes ne sont pas parfaites et
221                 # possèdent un point décalé des autres
222                 fin+=1
223                 # Permet de sortir de la boucle while
224                 maxim=lignes[k][1]
225                 imini=k
226                 # Quand la courbe atteint son point minimum, on stock l'indice
227                 # et la valeur de ce point
228             elif mini1>lignes[k][1]:
229                 #On cherche le minimum juste avant le pic R
230                 mini1=lignes[k][1]
231                 k-=1
232             else :
233                 k-=1
```

```
        k=imini-1
        # On redémarre au point minimum avant le pic
        while fin==1 and k!=0:
            if maxim>=lignes[k][1] and maxim>=lignes[k-1][1]:
                a=[lignes[k][0],lignes[k][1]]
                lQ.append(a)
                # Dès que la courbe remonte, on stocke les coordonnées de ce
                # point Q dans lQ
                fin+=1
            elif maxim<lignes[k][1]:
                maxim=lignes[k][1]
                k-=1
                # On cherche le maximum précédent le minimum trouvé
            elif maxim>=lignes[k][1] and lignes[k+1][1]==maxim:
                a=[lignes[k][0],lignes[k][1]]
                lQ.append(a)
                # Dès que la courbe remonte, on stock les coordonnées de ce
                # point Q dans lQ
                fin+=1
            else :
                k-=1
        mini2=lpics[i][2]
        k=lpics[i][1]+1
        # On démarre au point juste après le pic
        while fin==2 and k!=2000:
            if mini2<=lignes[k][1] and mini2<=lignes[k+1][1]:
                # Dès que la courbe remonte, on stocke les coordonnées de ce
                # point S dans lS
                maxi=lignes[k][1]
                imaxi=k
                fin+=1
            elif mini2>lignes[k][1]:
                mini2=lignes[k][1]
                k+=1
            else :
                k+=1
```

```
271
272         k=imaxi+1
273         # On démarre au point minimum après le pic
274         while fin==3 and k!=2000:
275             if maxi>=lignes[k][1] and maxi>=lignes[k+1][1]:
276                 b=[lignes[k][0],lignes[k][1]]
277                 lS.append(b)
278                 fin+=1
279             elif maxi>=lignes[k][1]:
280                 b=[lignes[k][0],lignes[k][1]]
281                 lS.append(b)
282                 fin+=1
283             elif maxi<lignes[k][1]:
284                 maxi=lignes[k][1]
285                 k+=1
286             else :
287                 fin+=1
288     for i in range (len(lS)):
289         lQRS.append(lS[i][0]-lQ[i][0])
290     if lQRS ==[]:
291         return 0
292     else:
293         QRS=sum(lQRS)/len(lQRS)
294         return QRS
295
```

```python
def défibrillateur(nom):
    print(nom,':')
    bpm,lpics=périodebpm(nom)
    print('bpm=',bpm)
    QRS=tailleQRS(nom)
    print('Taille QRS=',QRS,'\n')
    if bpm==0 and QRS==0: # cas d'une asystolie
        return False
    elif 50<bpm<180 and 0.07<QRS<0.1: # cas d'un patient en bonne santé
        return False
    elif bpm>=180: # cas d'une tachycardie ou fibrillation ventriculaire
        return True
    elif QRS>=0.1: # cas d'un QRS large
        return True
    else :
        return False
```