Détection de chute par accéléromètre

Numéro d'inscription : 36161

1 Introduction

2 Elaboration d'un accéléromètre capacitif

3 Expérimentation

4 Algorithme de détection des chutes et mise en application

5 Conclusion

6 Annexe

Contexte :

- Vieillissement & sédentarité $\rightarrow$ hausse du nombre de chutes mortelles
- Plan antichute personnes âgées lancé en février 2022 par l'actuelle ministre de la Santé alors en charge de l'Autonomie `https://solidarites-sante.gouv.fr/IMG/pdf/dp_plan-antichute-accessible28-02-2022.pdf`

Problématique :

Comment l'utilisation d'un accéléromètre peut permettre de détecter des chutes notamment chez les personnes âgées ?

Objectifs :

- Elaborer un accéléromètre capacitif
- Expérimenter ce capteur
- Traiter les données
- Développer un algorithme de détection des chutes

1 Introduction

2 Elaboration d'un accéléromètre capacitif

3 Expérimentation

4 Algorithme de détection des chutes et mise en application

5 Conclusion

6 Annexe

Pourquoi un accéléromètre capacitif ?

	Capacitif[1]	Piézoélectrique[2]
Amplitude	$\pm$400 g	$\pm$ 50.000 g
Fréquence de vibration	0 Hz à 100 Hz	2 à 25.000 Hz

	Décélération chute corps mou
Amplitude	$\pm$7 g
Fréquence de vibration	$\simeq 10Hz$

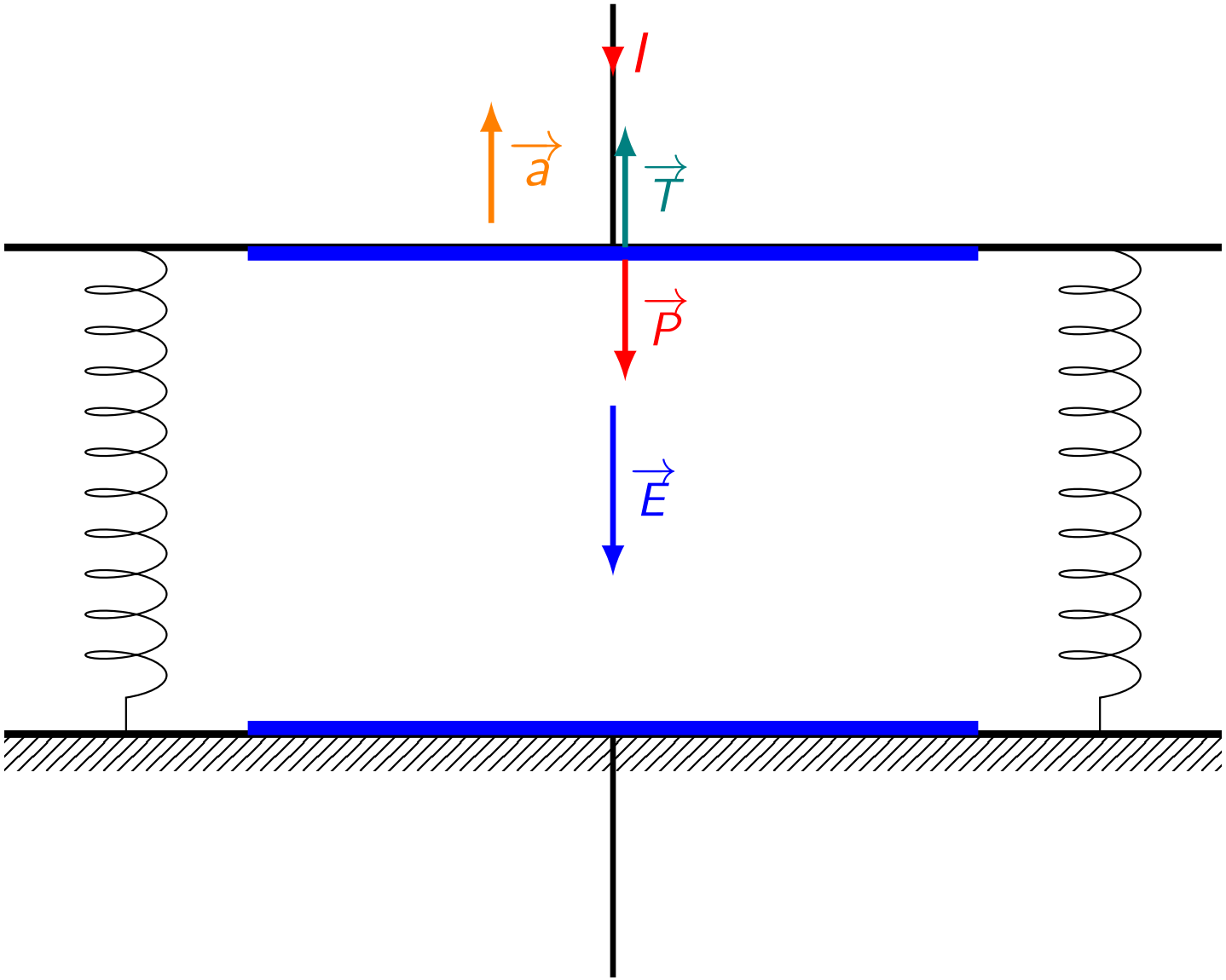
Table: Valeurs relevées avec Phyphox (cf Annexe)

Avantages de l'accéléromètre capacitif

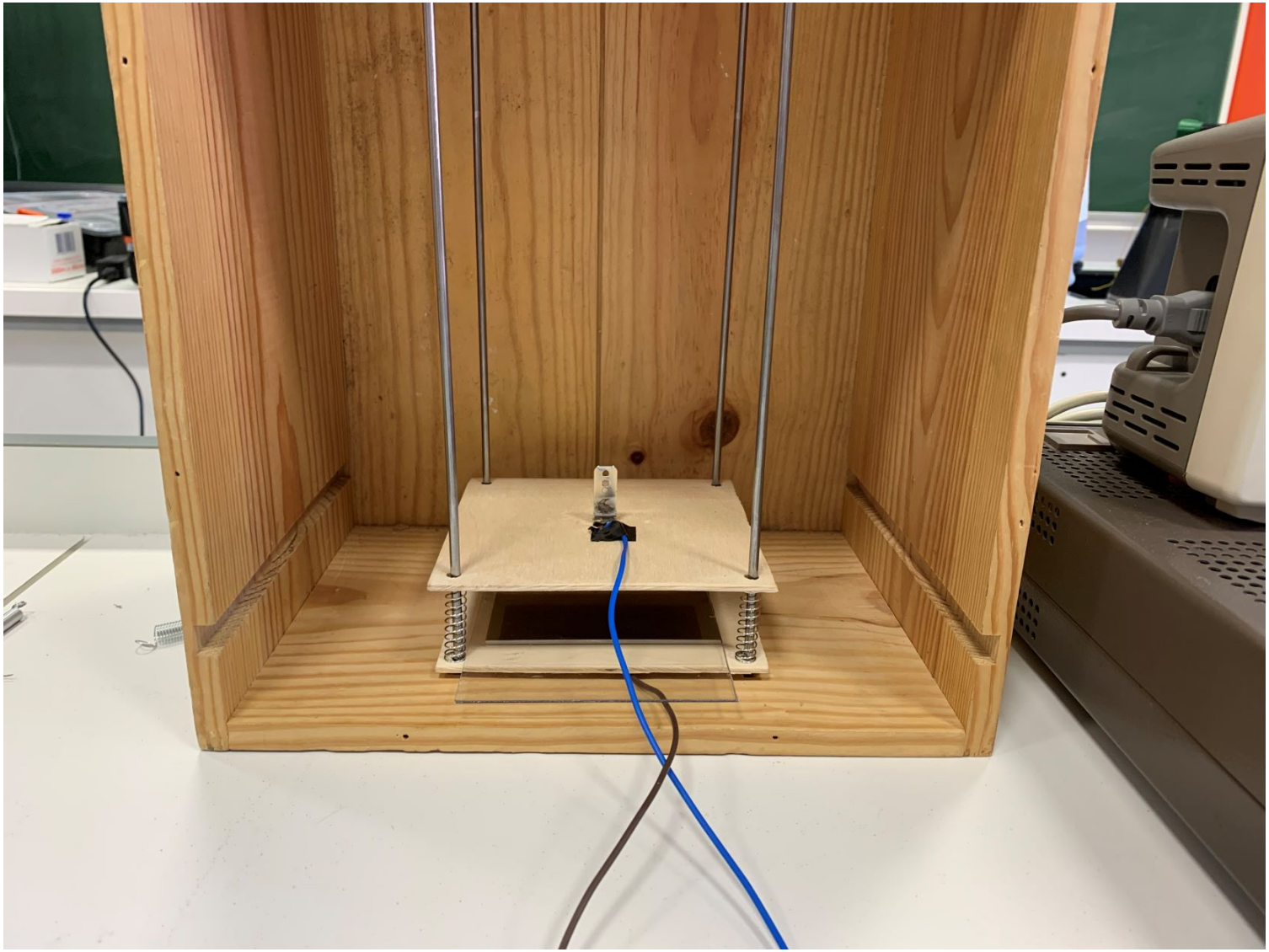
- plus résistant aux chocs
- plus simple à mettre en oeuvre
- parfaitement adapté à la chute de corps mous

[1] 411LN de ASC
[2] 350D02 de PCB PIEZOTRONICS

Oscillateur harmonique & condensateur variable



Accéléromètre fabriqué



plaque aluminium : 100 $\times$ 100 $\times$ 1 mm

Hypothèses du condensateur

- armatures parallèles, infinies et chargées uniformément
- insensible aux ondes électromagnétiques et à l'induction électromagnétique

Hypothèses de l'oscillateur

- liaisons parfaites
- pas de forces de frottement
- même constante de raideur pour chaque ressort

Equations à la base de l'étude

$$ma = -k(\ell - \ell_{eq}) \quad \text{avec} \quad \ell_{eq} = \ell_0 - \frac{m}{K}g \tag{1}$$

$$C = \frac{\varepsilon_0 \varepsilon_r S}{\ell} \quad \text{avec} \quad \varepsilon_r = 3,3 \tag{2}$$

1 Introduction

2 Elaboration d'un accéléromètre capacitif

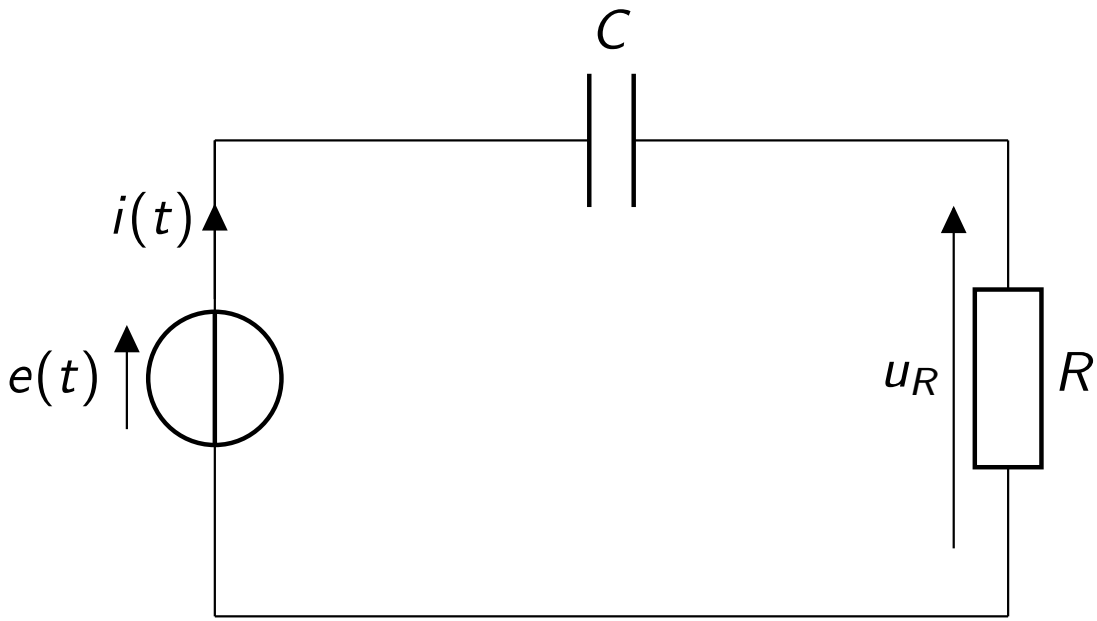
3 Expérimentation

4 Algorithme de détection des chutes et mise en application

5 Conclusion

6 Annexe

Expérience 1 : Mesure en circuit RC (entrée créneau)



Expression de la capacité

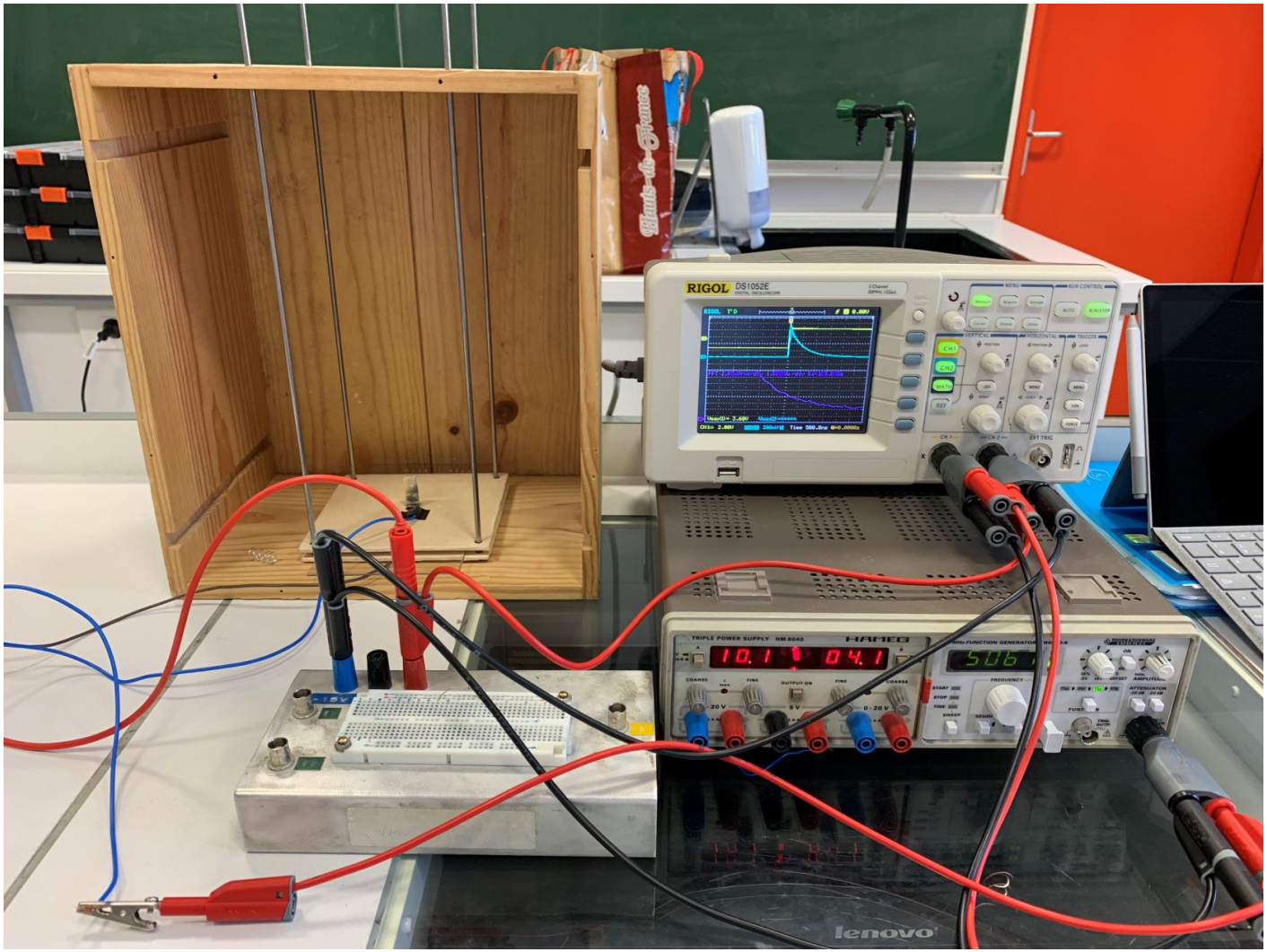
$$\frac{d^2 i}{dt^2} + \frac{1}{RC}\frac{di}{dt} = 0$$

$$u_R(t) = RI_0 \exp\left(\frac{-t}{RC}\right)$$

On cherche à déterminer expérimentalement la constante de temps RC

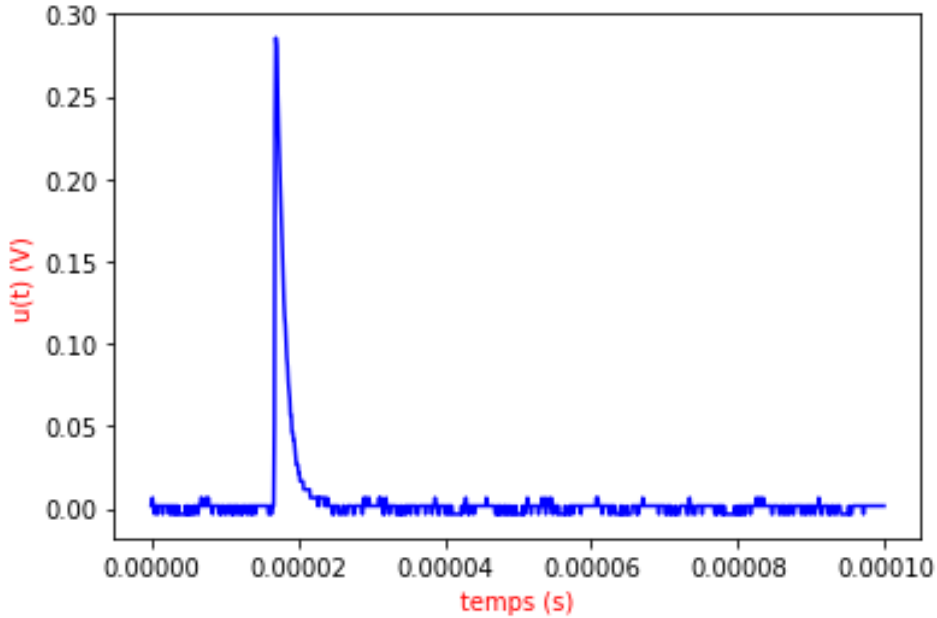
Montage et réglages

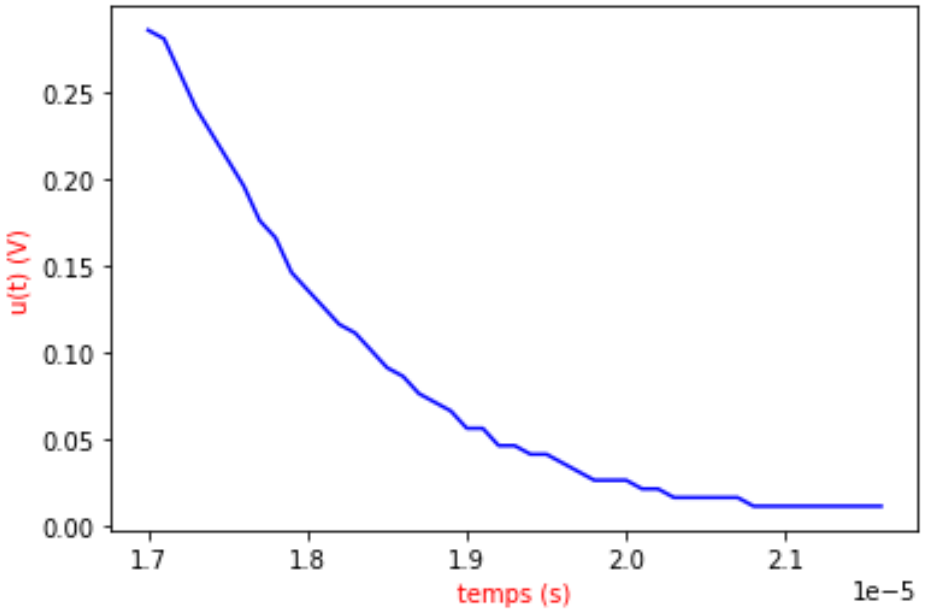
- entrée créneau
- décharge condensateur $\rightarrow$ décroissance exponentielle



Résultats

- Acquisition avec boîtier Sysam-SP5 sous Latis-Pro
- Traitement des données sous Python (cf Annexe)

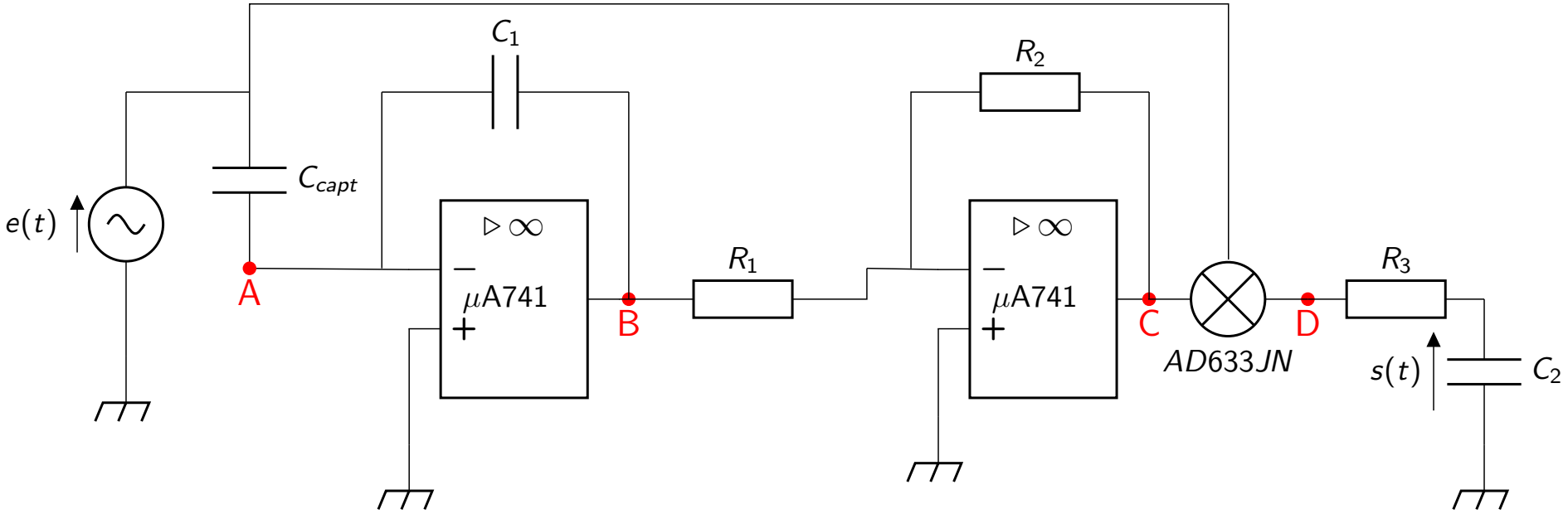




Mesure de la capacité avec ln(u(t))

	Capacité
e = 1 mm	C = 150 pF
e = 3,1 mm	C= 40 pF

Expérience 2 → tension de sortie continue proportionnelle à C_{capt}



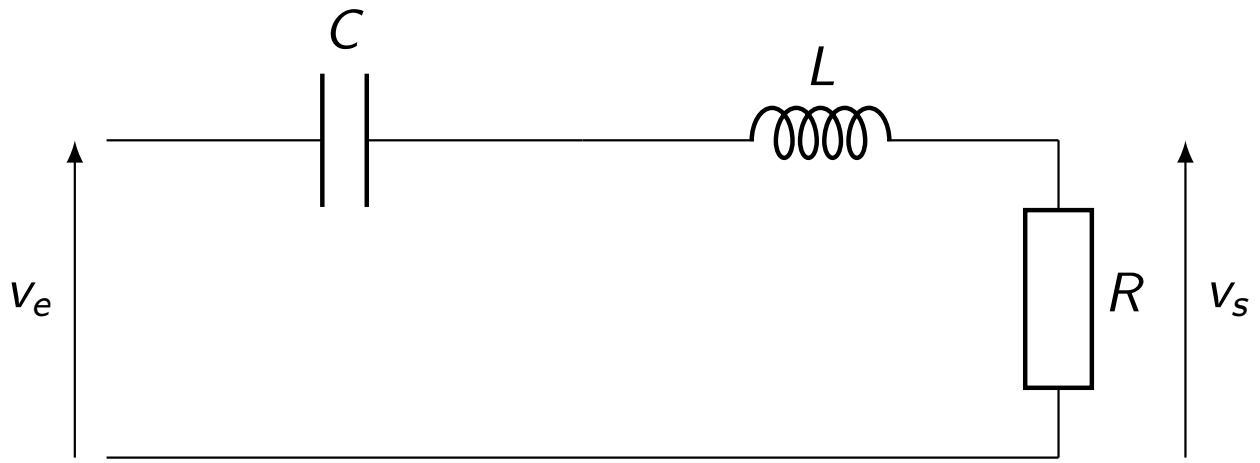
Principe, hypothèses et équation finale

- Rapport des capacités / Amplificateur inverseur / Démodulateur synchrone
- Déphasage nul entre l'entrée et la sortie
- ALI idéal de gain infini / multiplieur parfait

$$U_{C_2} = 0,1 \cdot \frac{A^2}{2} \frac{R_2 C_{capt}}{R_1 C_1}$$

Ajout d'un passe-bande en sortie du 2^e *ALI*

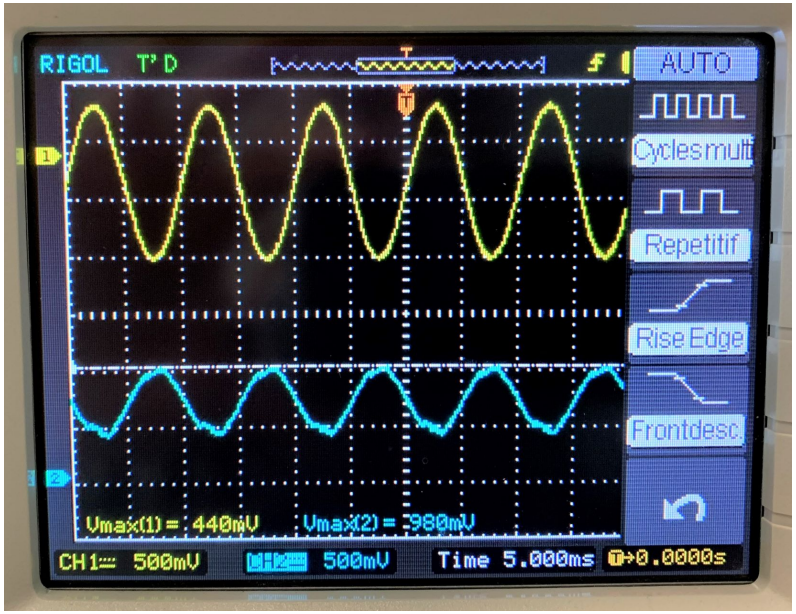
- Supprimer les artefacts du GBF
- Supprimer une composante continue due à une dérive en amont du multiplieur

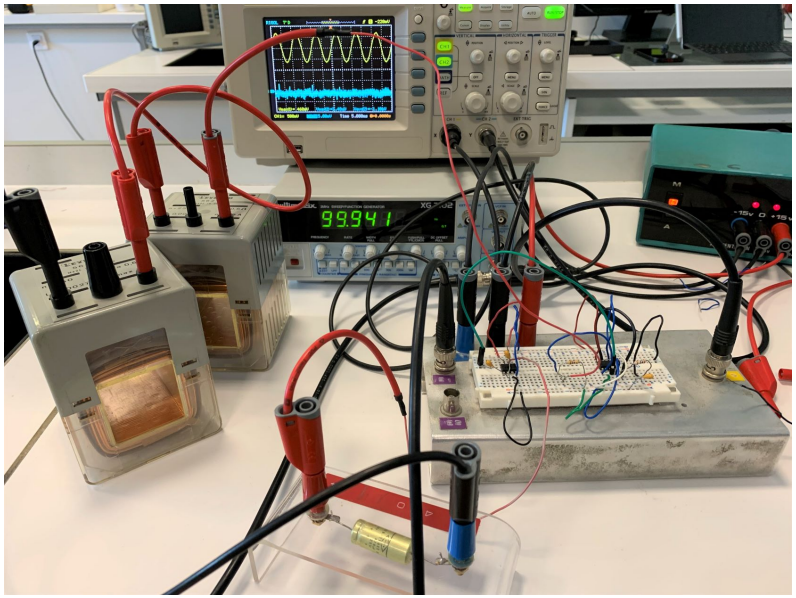


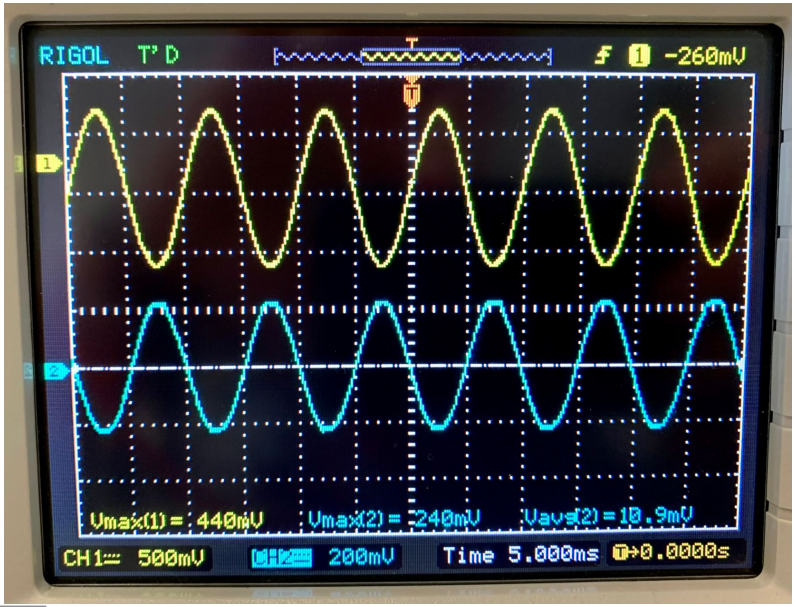
Résultats

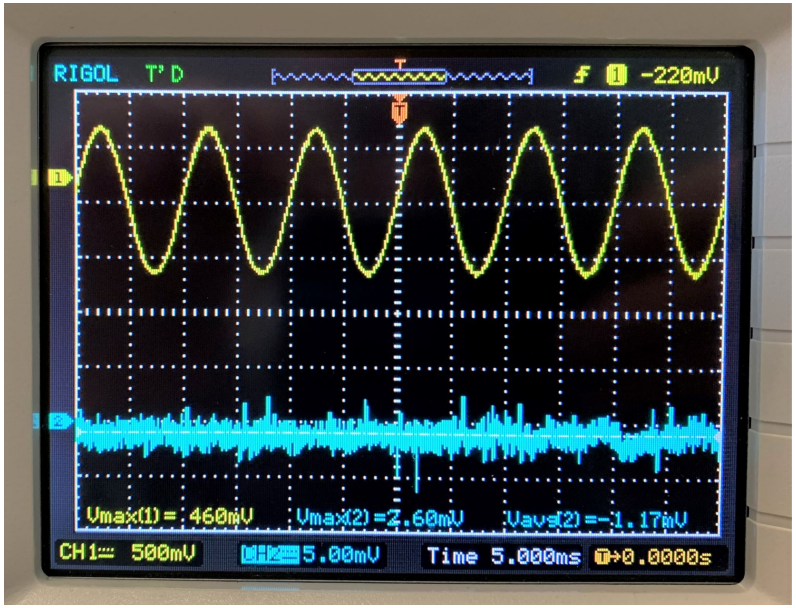
- Amélioration de la sinusoïde
- Suppression de la composante continue
- Signal de sortie théorie $= -9,68\,mV$ / expérience $= -1,17\,mV$

Résultats


RIGOL T'D
AUTO
Cyclesmulti
Repetitif
Rise Edge
Frontdesc
Vmax(1)= 440mV Vmax(2)= 980mV
CH1 500mV 500mV Time 5.000ms 0.0000s




RIGOL T'D
-260mV
Vmax(1)= 440mV Vmax(2)= 240mV Vavs(2)= 10.9mV
CH1 500mV 200mV Time 5.000ms 0.0000s


RIGOL T'D
-220mV
Vmax(1)= 460mV Vmax(2)=2.60mV Vavs(2)=-1.17mV
CH1 500mV 5.00mV Time 5.000ms 0.0000s

1 Introduction

2 Elaboration d'un accéléromètre capacitif

3 Expérimentation

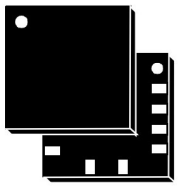
4 Algorithme de détection des chutes et mise en application

5 Conclusion

6 Annexe

1) Approche simplifiée de la détection de chutes

Matériel : micro:bit v2.0 → LSM303AGR de STMicroelectronics



LGA-12 (2.0x2.0x1.0 mm)

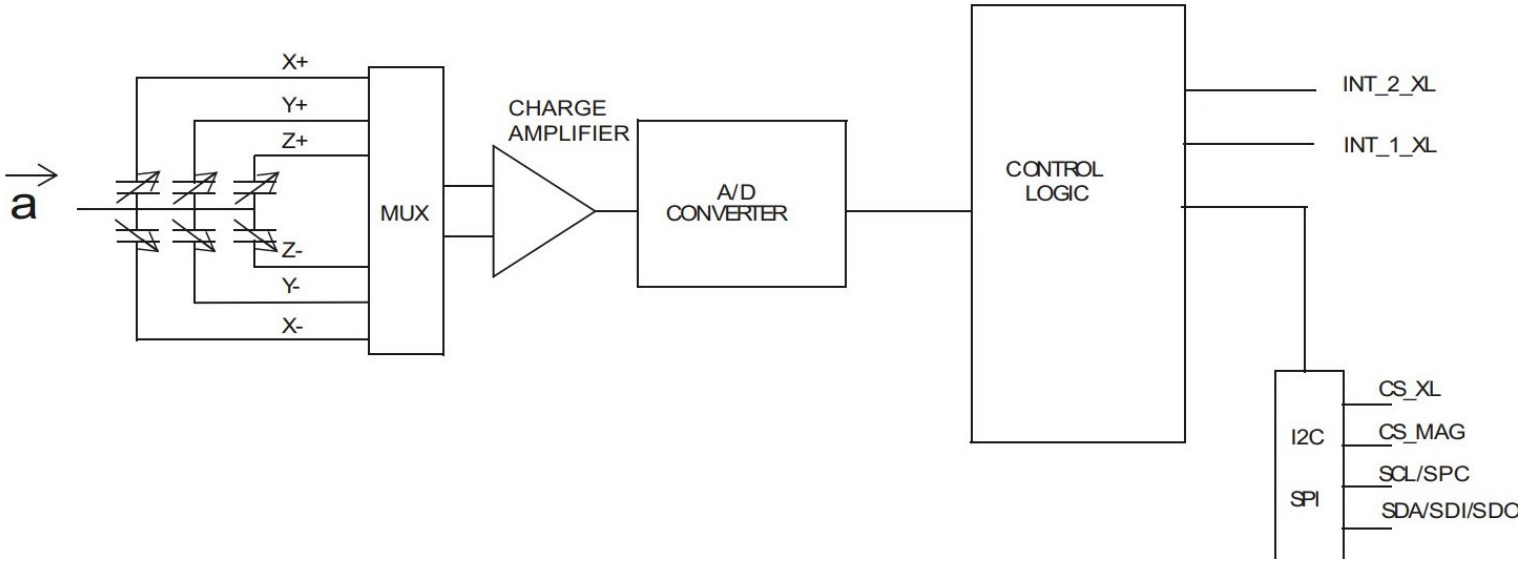
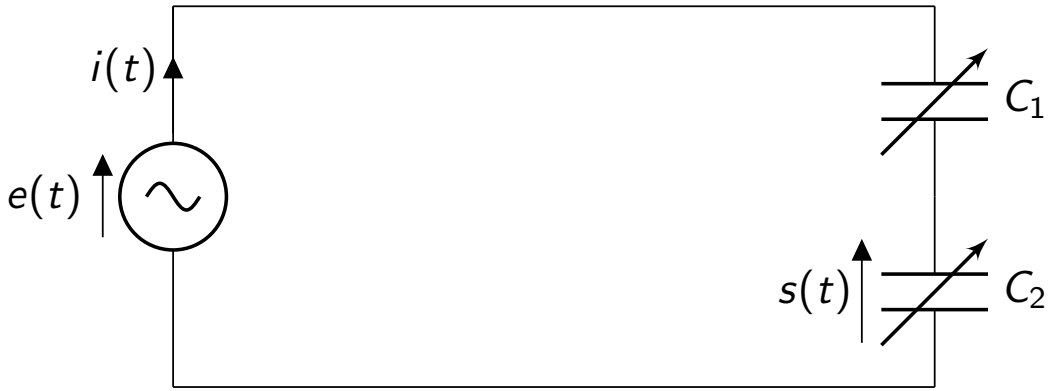


Figure: STMicroelectronics

Montage double différentiel

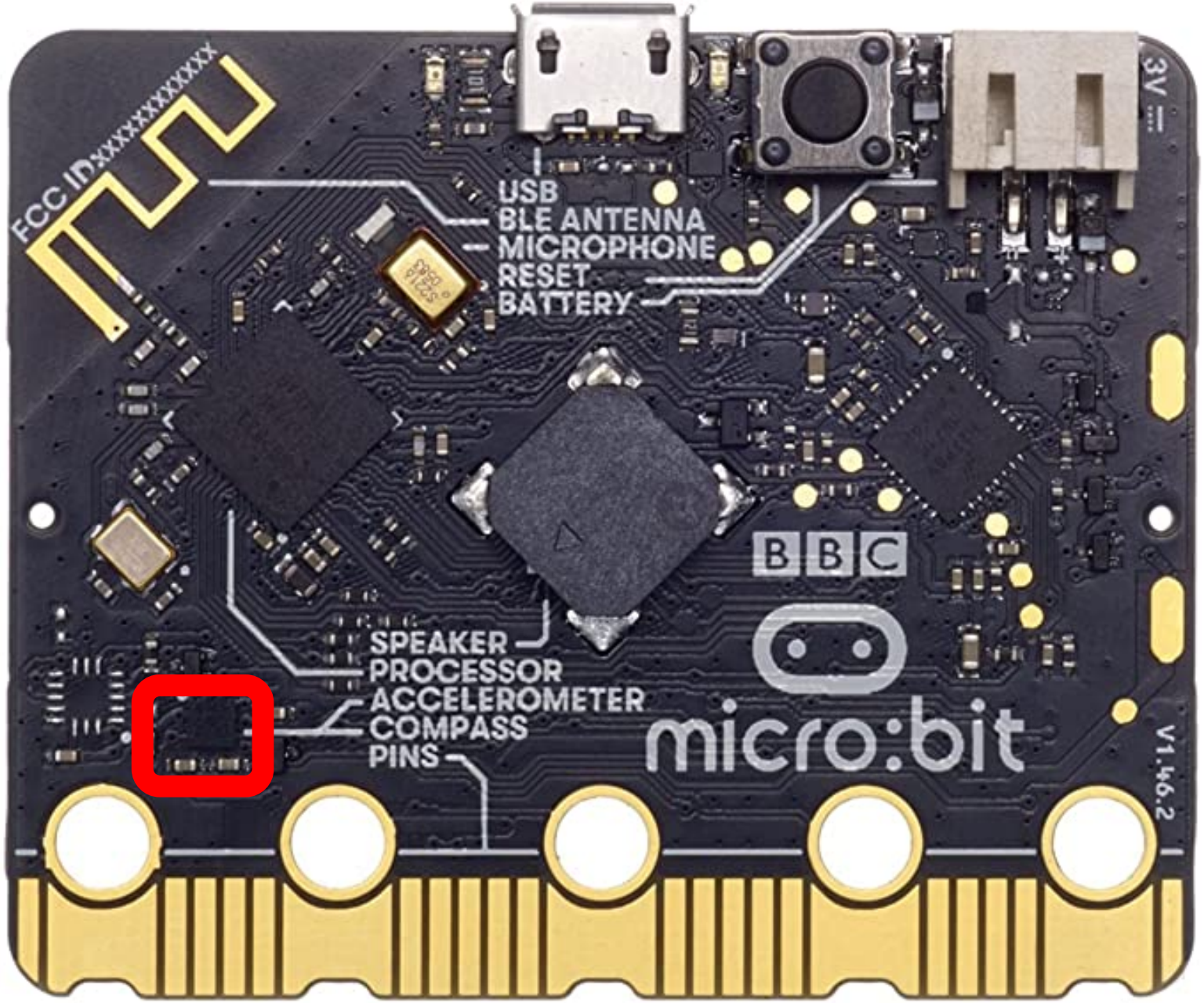


Avantages montage double différentiel

- Variation linéaire de la capacité

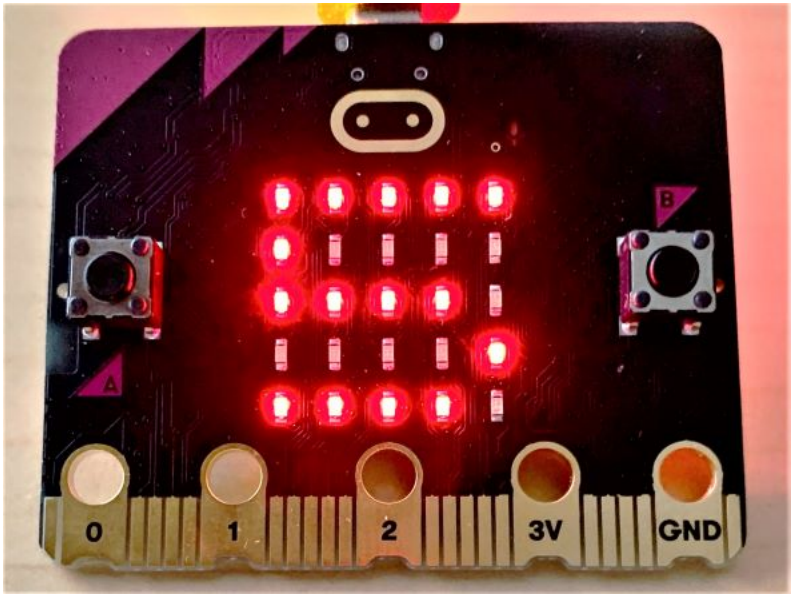
$$\underline{s} = \frac{Z_{C2}}{Z_{C1}}\underline{e} = \frac{1}{2}\left(1 + \frac{\ell}{D_0}\right)\underline{e}$$

- Montage push-pull $\rightarrow$ compensation des grandeurs d'influence (cf Annexe)

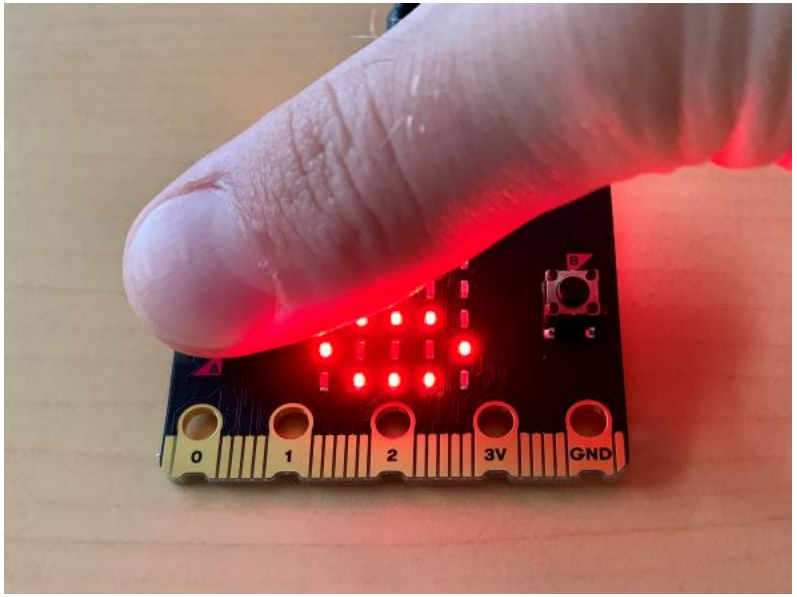

FCC ID:xxxxxxxxxx
USB
BLE ANTENNA
MICROPHONE
RESET
BATTERY
3V
SPEAKER
PROCESSOR
ACCELEROMETER
COMPASS
PINS
BBC
micro:bit
V1.46.2

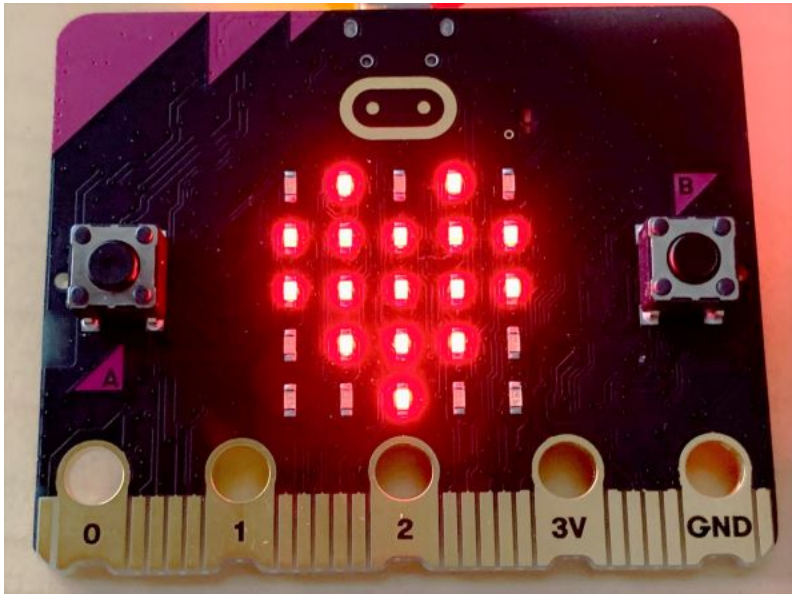
© micro:bit

Images du fonctionnement



- Détection chute → lancement décompte de 10 s
- Si appui bouton gauche → arrêt du décompte et affichage coeur
- Sinon après décompte → déclenchement alarme





1 Introduction

2 Elaboration d'un accéléromètre capacitif

3 Expérimentation

4 Algorithme de détection des chutes et mise en application

5 Conclusion

6 Annexe

Conclusion :

- Accéléromètre capacitif $\rightarrow$ capteur adapté
- Précision du capteur n'est pas rédhibitoire
- Algorithme de seuil suffisament performant

Pistes à approfondir :

- Etude statistique $\rightarrow$ évaluer performances du capteur
- Développer un algorithme machine learning $\rightarrow$ plus de fiabilité et d'aptabilité

1 Introduction

2 Elaboration d'un accéléromètre capacitif

3 Expérimentation

4 Algorithme de détection des chutes et mise en application

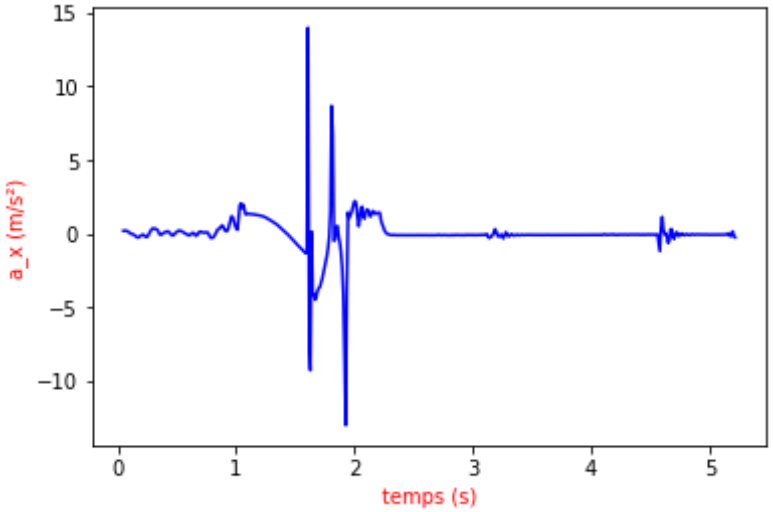
5 Conclusion

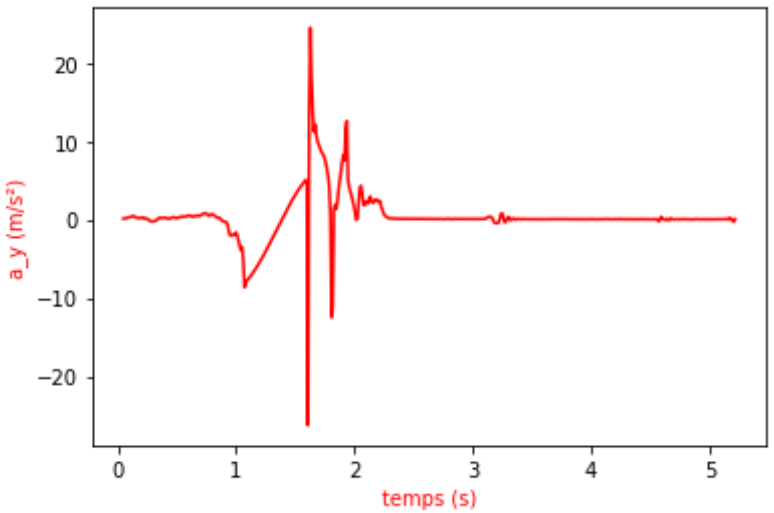
6 Annexe

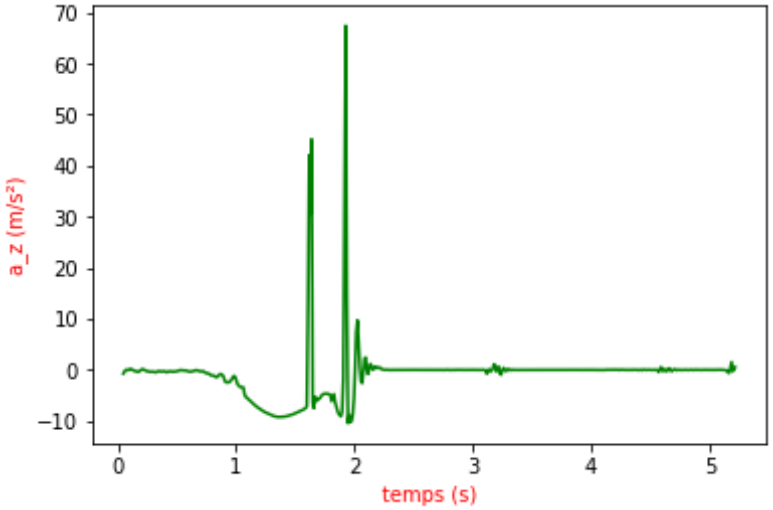
Phyphox (1/2)

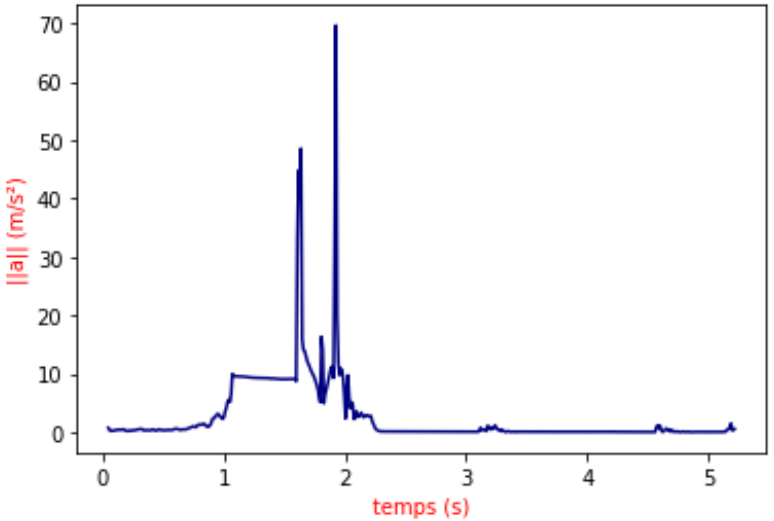
- Vibration $\sim$ 10 Hz
- Accélération max $=$ 7 g

Phyphox (2/2)









Code Python pour le traitement des données (1/2)

```
# -*- coding: utf-8 -*-

import csv
import numpy as np
import matplotlib.pyplot as plt


with open('C:\\Users\\arthu\\Desktop\\Circuit RC TIPE 3.csv', newline='',encoding='utf-8') as f:
    data = np.genfromtxt('C:\\Users\\arthu\\Desktop\\Circuit RC TIPE 3.csv', delimiter=';')

data = np.delete(data,0,0)    #supprimer les noms des colonnes du fichier csv

X1 = [data[x][0] for x in range(len(data))]
Y1 = [data[y][1] for y in range(len(data))]
plt.plot(X1,Y1,color='b')
plt.xlabel("temps (s)", color ='r')
plt.ylabel("u(t) (V)", color = 'r')
plt.show()

def tab_reduit(T): #pour réduire aux valeurs intéressantes
    i = 0
    while T[i][1] < 0.01 :
        i += 1
    j = i
    while T[j][1] > 0.01 :
        j += 1
    new_tab = T[i:j]
    return new_tab

def indice_max(T):
    max = 0
    for i in range(len(T)):
        if T[i][1] > max:
            max = T[i][1]
            j = i
    return j

def tracé_expérimental(T):   #limiter l'étude à la décroissance exponentielle
    reduit = tab_reduit(T)   #afin de calculer la constante de temps
    j = indice_max(reduit)
```

Code Python pour le traitement des données (2/2)

```
    return reduit[j:]

tracé = tracé_expérimental(data)
X2 = [tracé[x][0] for x in range(len(tracé))]
Y2 = [tracé[y][1] for y in range(len(tracé))]
plt.plot(X2,Y2,color='b')
plt.xlabel("temps (s)", color ='r')
plt.ylabel("u(t) (V)", color = 'r')
plt.show()

def ajustement(T): #Ajuster log(courbe) par un polynôme d'ordre 1
    tracé = tracé_expérimental(T)
    X = [tracé[x][0] for x in range(len(tracé))]
    Y = [tracé[y][1] for y in range(len(tracé))]
    return np.polynomial.polynomial.polyfit(X,np.log(Y),1) #renvoie [b,a] de y = ax + b

ajustée = ajustement(data)

R = 80 #Ohms
capacité = -1/(80*10**(-2)*ajustée[1])

print(ajustée)
print(capacité)
```

Expérience 2

Valeurs des dipôles utilisés

	R_1	$R_2 = R_3$	$C_1 = C_2 = C_{capt}$
Valeur	10 Ω	10 kΩ	1 μF

Equations

$$V_B = -\frac{C_{capt}}{C_1}$$

$$V_C = -\frac{R_2}{R_1} V_B = AB sin(\omega t) \text{ avec } B = A\frac{R_2 C_{capt}}{R_1 C_1}$$

$$V_D = 0,1 \times V_C \times e = 0,1 \times \frac{AB}{2} cos(\Phi) - 0,1 \times \frac{AB}{2} cos(4\pi f \phi t + \Phi)$$

Déphasage nul $\Rightarrow \Phi = 0$

$$s(t) = 0,1 \times \frac{AB}{2}$$

Push-pull : compensation des grandeurs d'influence

Situation initiale :

$$m = m_0 \quad ; \quad g = g_0 \quad ; \quad Z_{C10} = Z_{C20} = Z_{C0} \quad ; \quad \underline{s} = \frac{\underline{e}}{2}$$

Après variation du mesurande :

$$Z_{C1} = Z_{C0} + \Delta Z_{C1} \quad ; \quad \Delta Z_{C1} = S_g \Delta g - S_m \Delta m$$

$$Z_{C2} = Z_{C0} + \Delta Z_{C2} \quad ; \quad \Delta Z_{C2} = S_g \Delta g + S_m \Delta m$$

$$\underline{s} + \Delta \underline{s} = \frac{Z_{C2}}{Z_{C1} + Z_{C2}} \underline{e}$$

$$\Delta \underline{s} = \frac{\underline{e}}{2Z_{C0}} \cdot \frac{S_m \Delta m}{1 + \frac{S_g \Delta g}{R_{C0}}}$$

* Sans push-pull $\rightarrow$ 4 au lieu de 2 au dénominateur

Sensibilité montage : $S = \frac{\Delta \underline{s}}{\Delta m}$

Algorithme de détection des chutes (1/2)

```
# Add your Python code here. E.g.
from microbit import *
import math

'''detection d'une chute à partir d'un objet connecté :
# 1/ Début de la chute : le corps est déséquilibré mais l'accélération du
corps ne change pas = 1g.

# 2/ Chute jusqu'au sol : jusqu'au contact sur le sol, l'accélération change
et l'angle d'arrêt et la vitesse angulaire de l'objet tourant autour de l'axe
 change.

# 3/ Impact : à partir du moment où le corps entre en contact avec le sol
jusqu'à ce qu'il s'arrête sur le sol, à ce stade, la valeur d'accélération
au milieu du corps changera radicalement et supportera l'impact maximal

# 4/ Immobilité : la phase post-impact est généralement caractérisée par le
fait de s'allonger sur le sol pendant un certain temps'''

# accelerometer.set_range(4)
CHUTE_LIMIT_G = 1.5;
CHUTE_AMPLITUDE_G = 1;
CHUTE_ANGLE_G = 60;
GRAVITY = 9.8

acceleration_max = 0;
acceleration_min = 9;
acceleration_min = 9;
#accelerometer.set_range(8)
isChute = 0;

while isChute == 0:

    x = accelerometer.get_x()
    y = accelerometer.get_y()
    z = accelerometer.get_z()
    v = accelerometer.get_values()

    presses = button_a.get_presses()
    acceleration = math.sqrt((x * x) + (y * y) + (z * z) )/1000
```

Algorithme de détection des chutes (2/2)

```
angle = math.acos(acceleration/GRAVITY)*180
acceleration_min = min(acceleration_min, acceleration)
acceleration_max = max(acceleration_max, acceleration)

print(v, acceleration, acceleration_min, acceleration_max, angle)

amp = acceleration_max-acceleration_min;

if acceleration > CHUTE_LIMIT_G:
    pressed = False
    for i in range(9, 0, -1):
        display.show(i)
        sleep(1000)
        if button_a.get_presses() > presses:
            pressed = True
            # display.show(" ")
            display.show(Image.HEART)
            break

    if not (pressed):
        print("help")
        audio.play(Sound.SPRING)
        display.show(" ")

sleep(50)
```