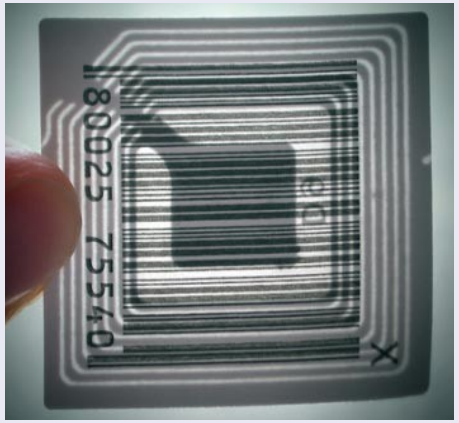
Communication des cartes à puce sans contact : principe et applications

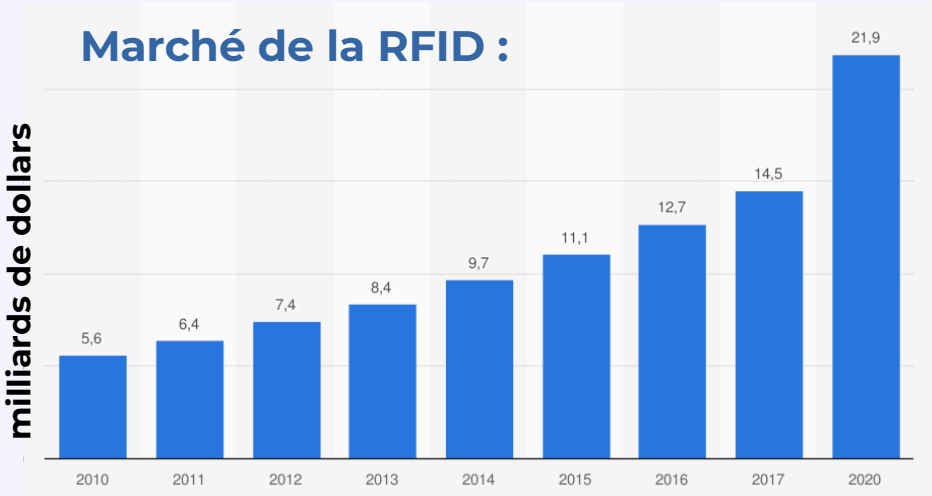
Thème de l'année : la ville

Enzo
MORAN
33934

Applications :



Nouveaux dispositifs :



Problématique :

Comment dialoguer et récupérer les informations contenues sur une carte de transport ?

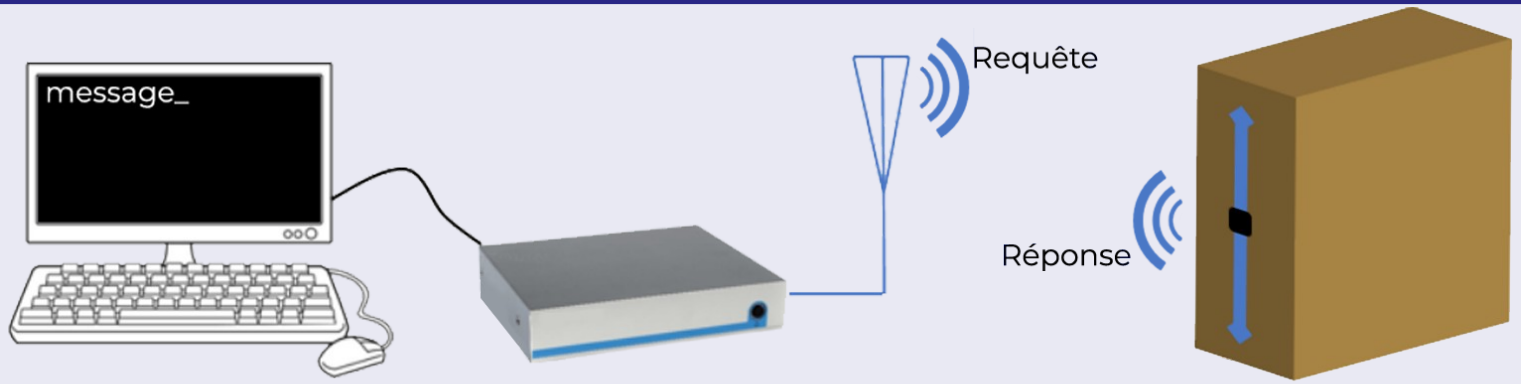
I – Communication par la technologie RFID

II – Modélisation de la liaison carte-lecteur

III – Transmission de notre propre message

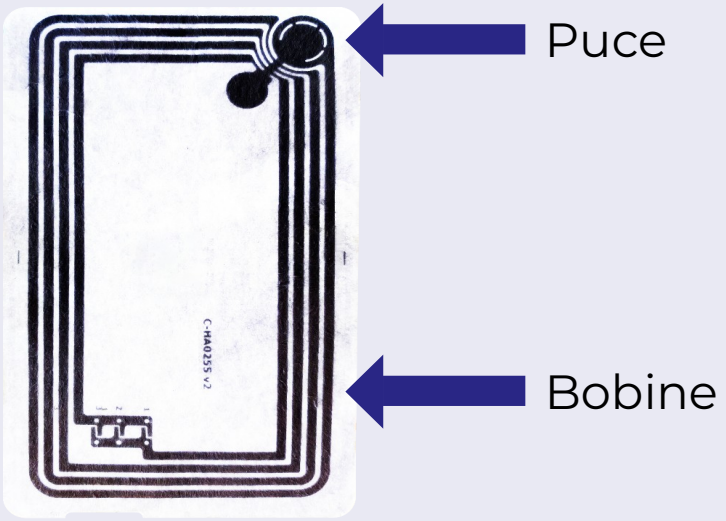
I / Communication par la technologie RFID

RFID : Radio Frequency Identification



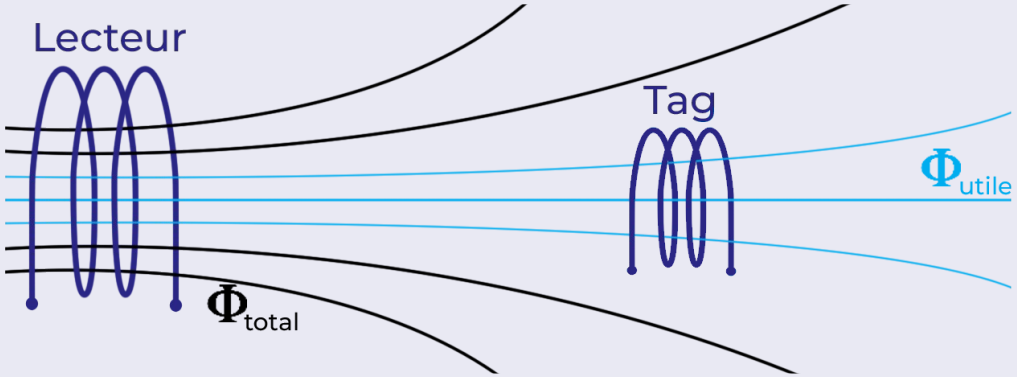
Lecteur :

Carte de bus :



I / Communication par la technologie RFID

PRINCIPE : Induction magnétique



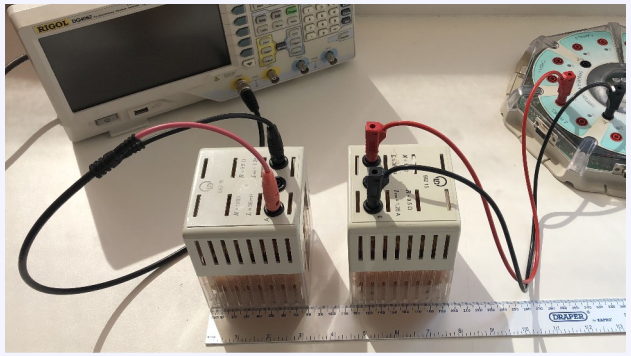
Loi de Faraday :

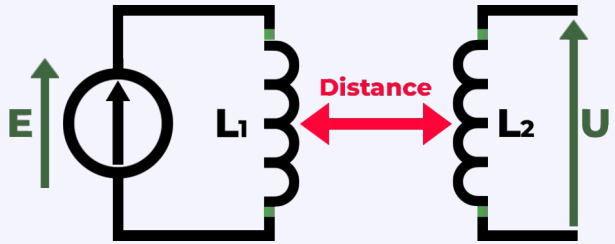
$$e = -\frac{\partial \Phi}{\partial t}$$

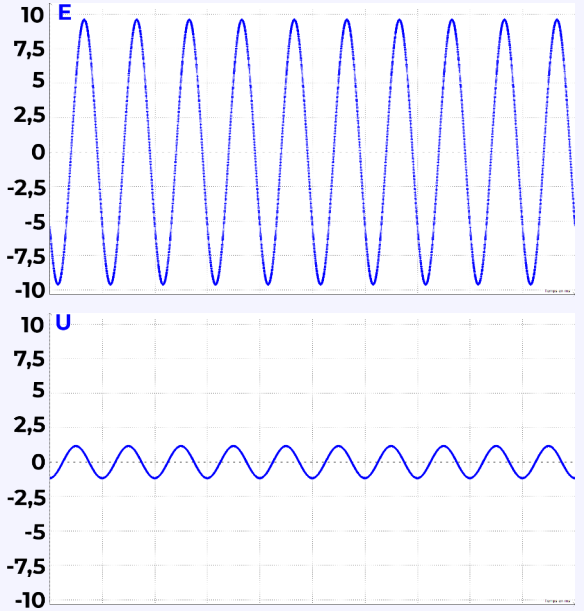
Flux :

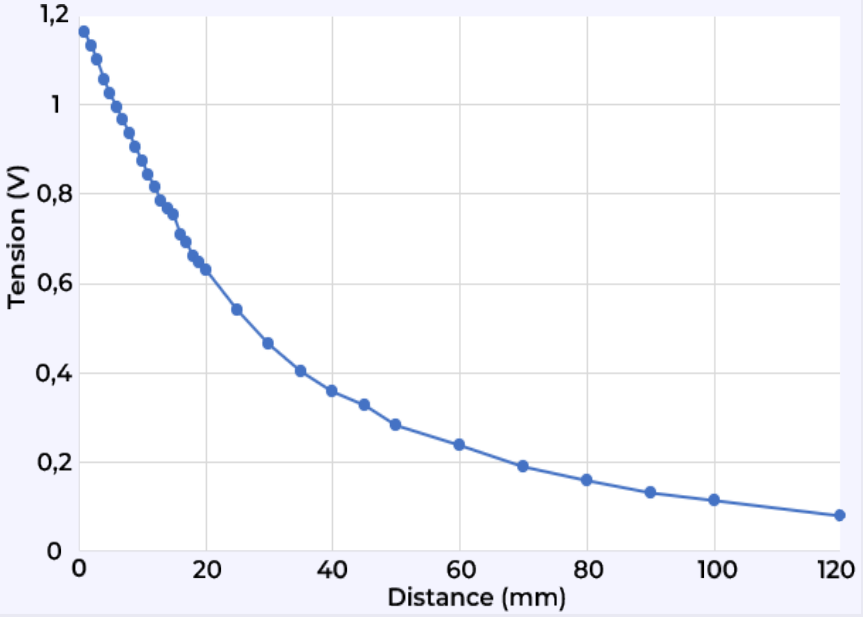
$$\Phi = \iint \vec{B} \cdot \overrightarrow{ds} = BNs$$

EXPÉRIENCE :



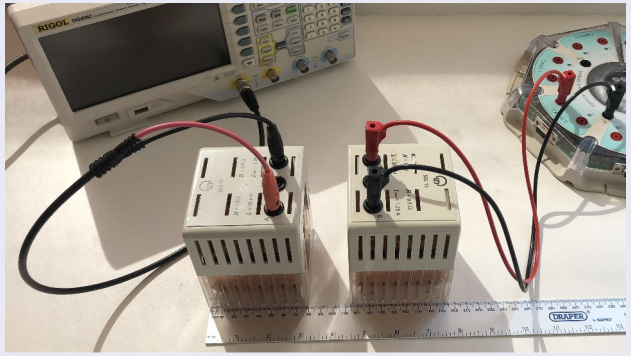


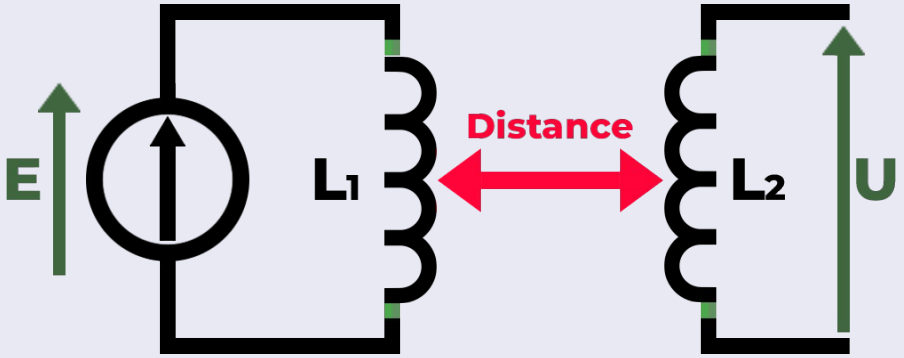




I / Communication par la technologie RFID

EXPÉRIENCE : Influence de la fréquence sur l'induction

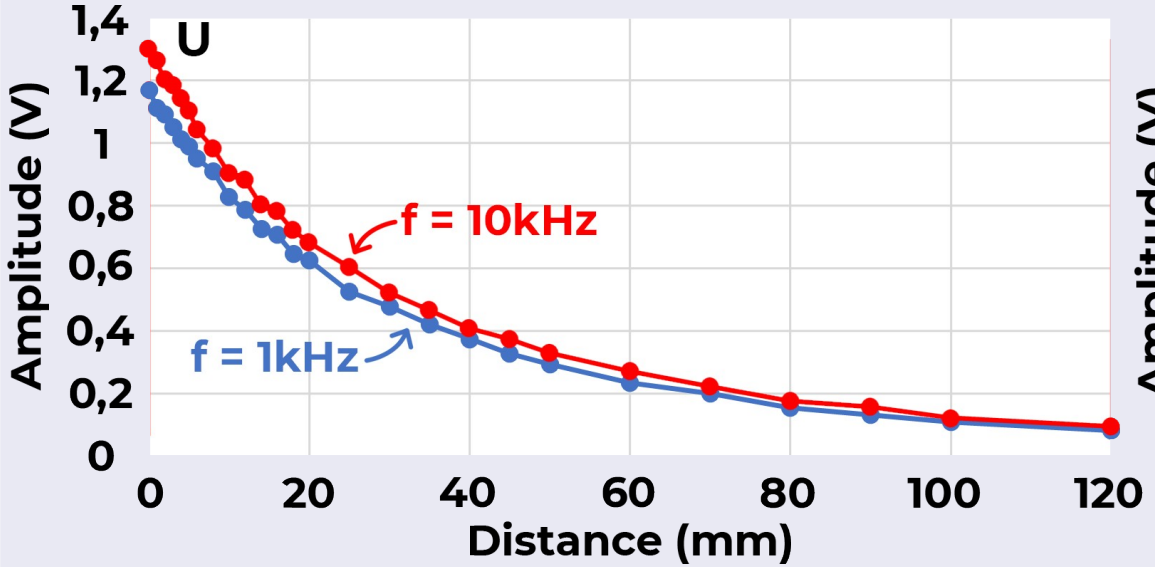


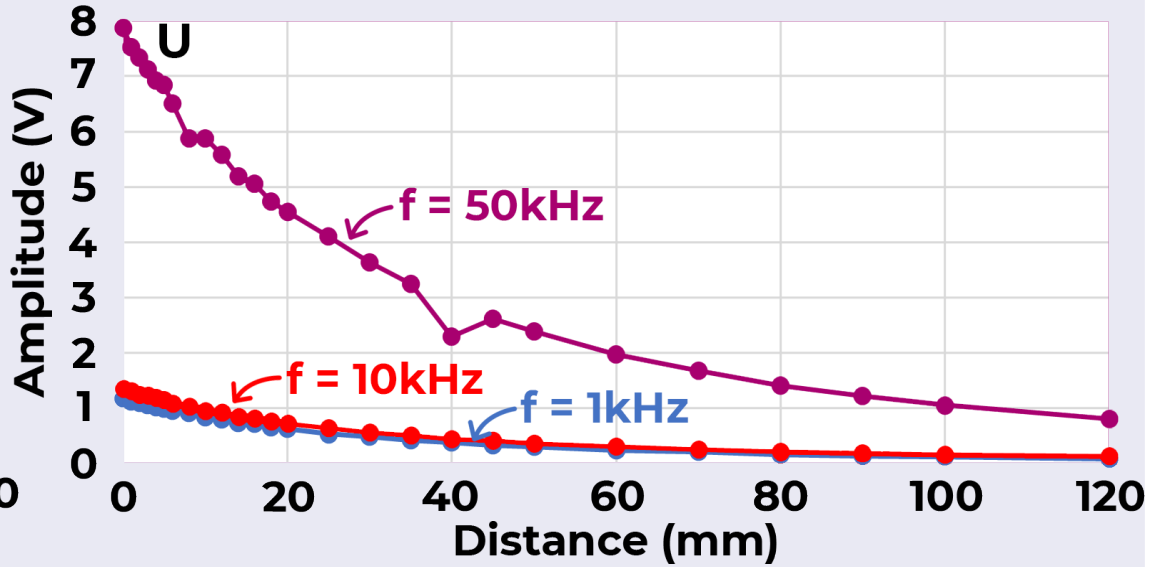


Signal :

Amplitude : **10V**

Fréquence : **1kHz**
10kHz
50kHz





I / Communication par la technologie RFID

FONCTIONNEMENT :

Gammes de fréquence :

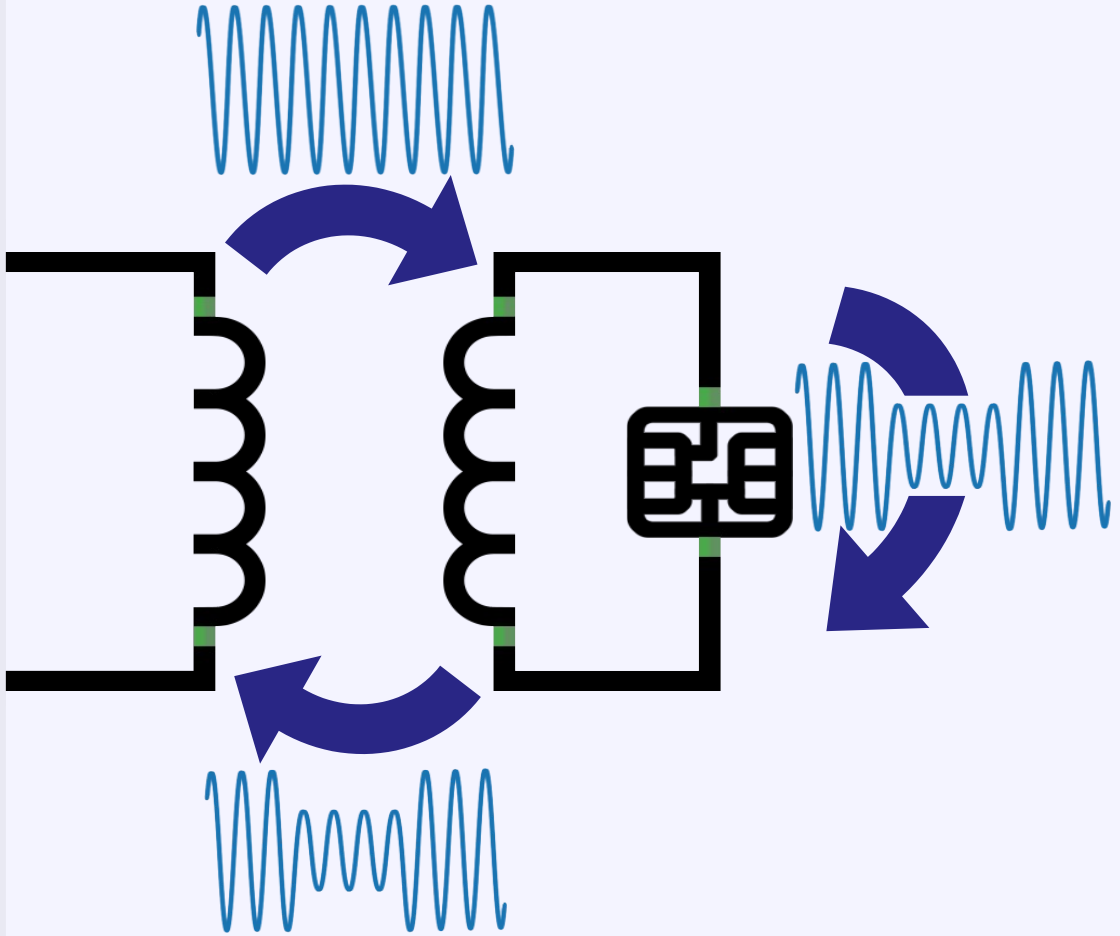
135 kHz : BF
cartes de cantine
identification animale

13,56 MHz : HF
titres de transport
paiements sans contact

5,8 GHz : UHF
péages routiers

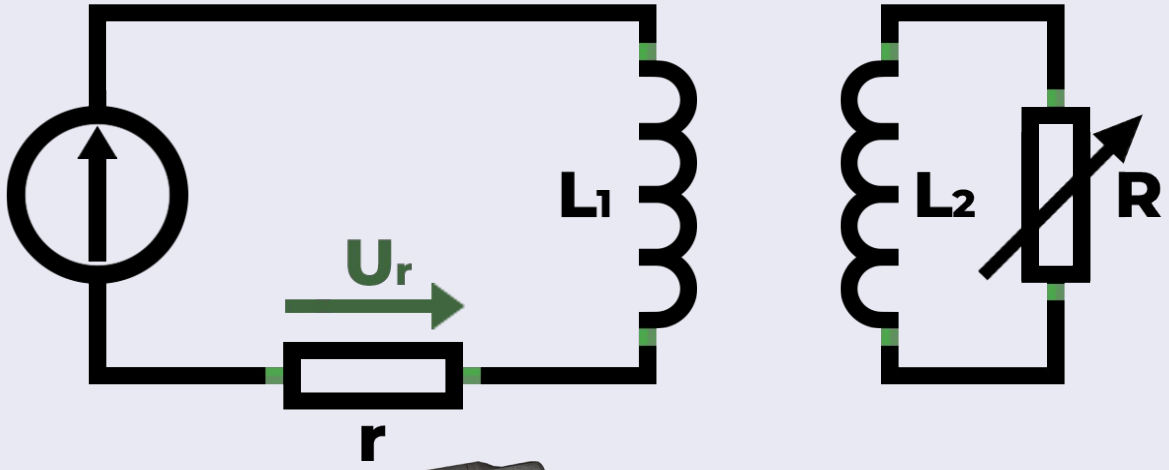
Couplage magnétique

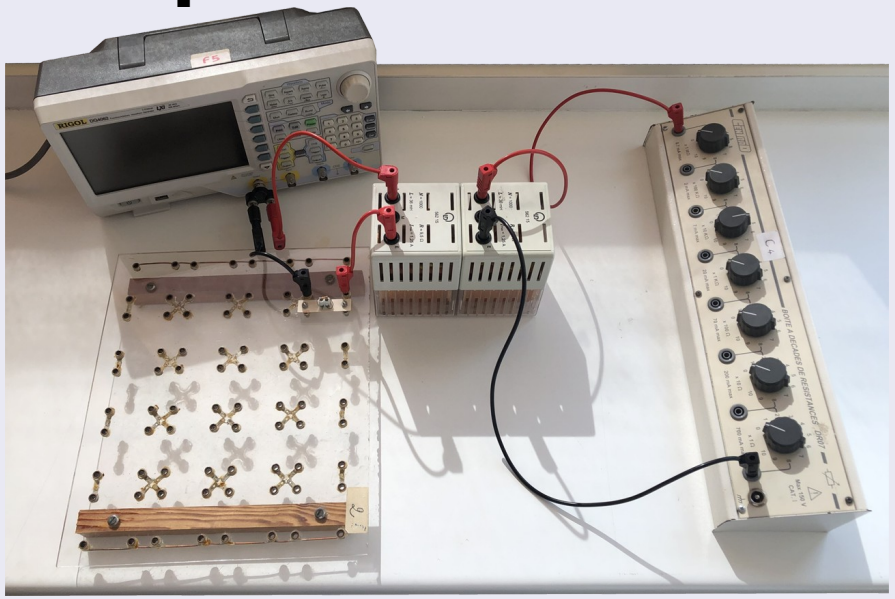
Modulation :

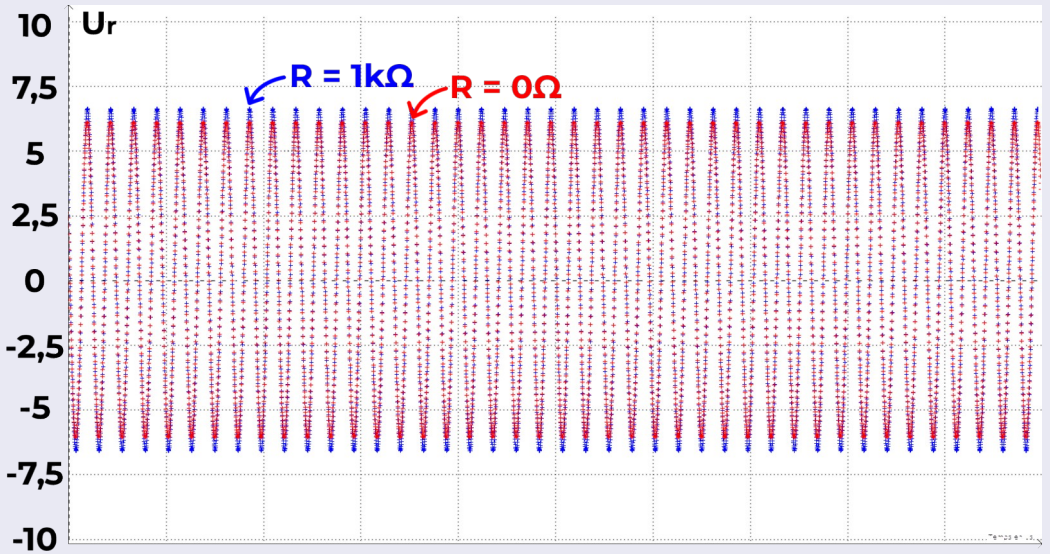


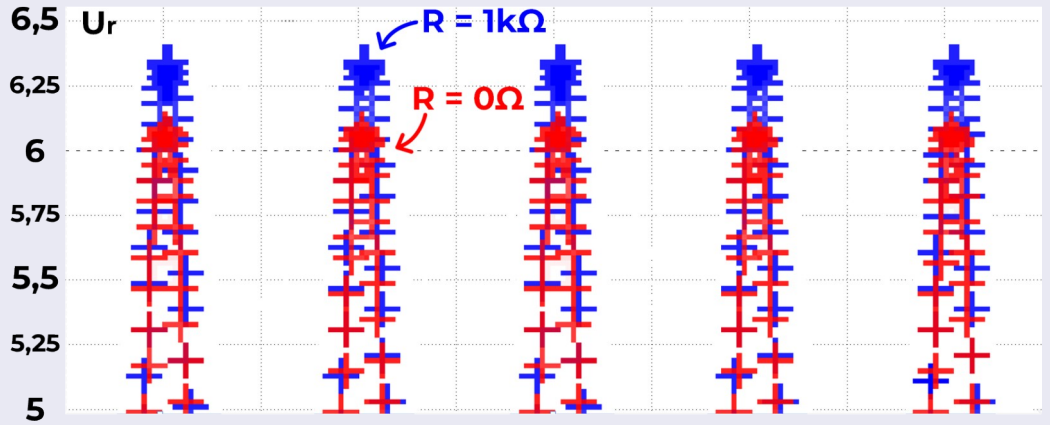
II / Modélisation de la liaison carte-lecteur

Modélisation initiale :



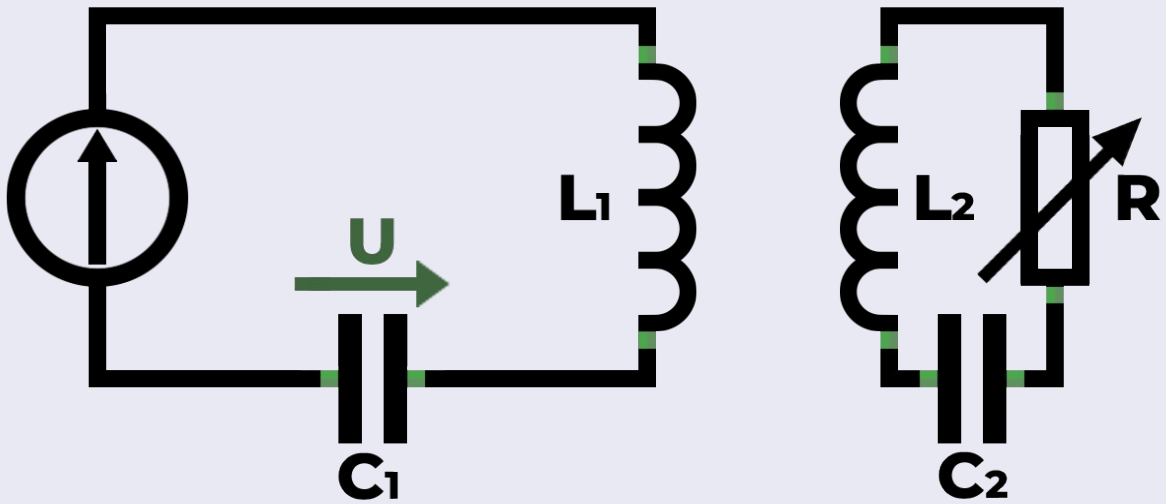


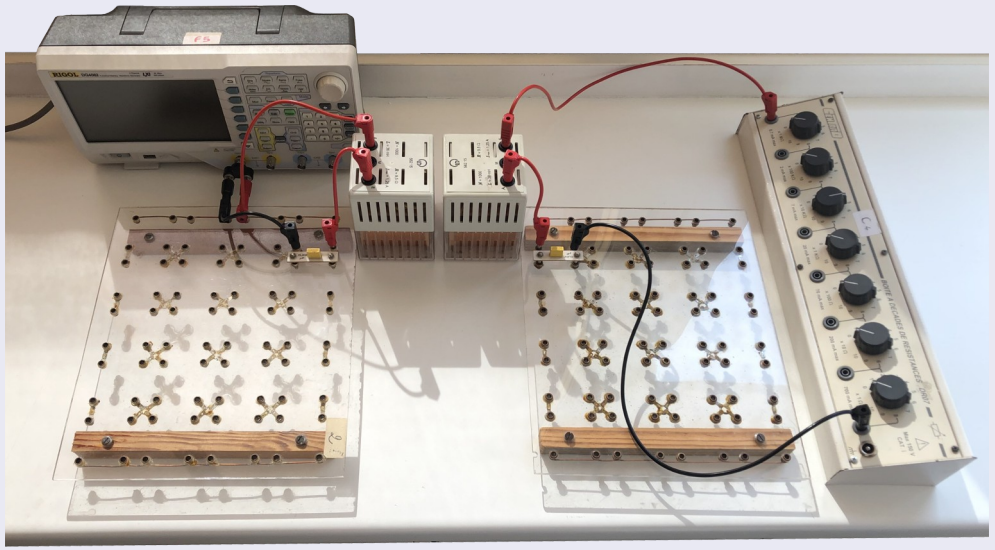




II / Modélisation de la liaison carte-lecteur

Modélisation finale :





Caractéristiques :

Lecteur :

- L_1 = 9 mH
- C_1 = 0,22 µF

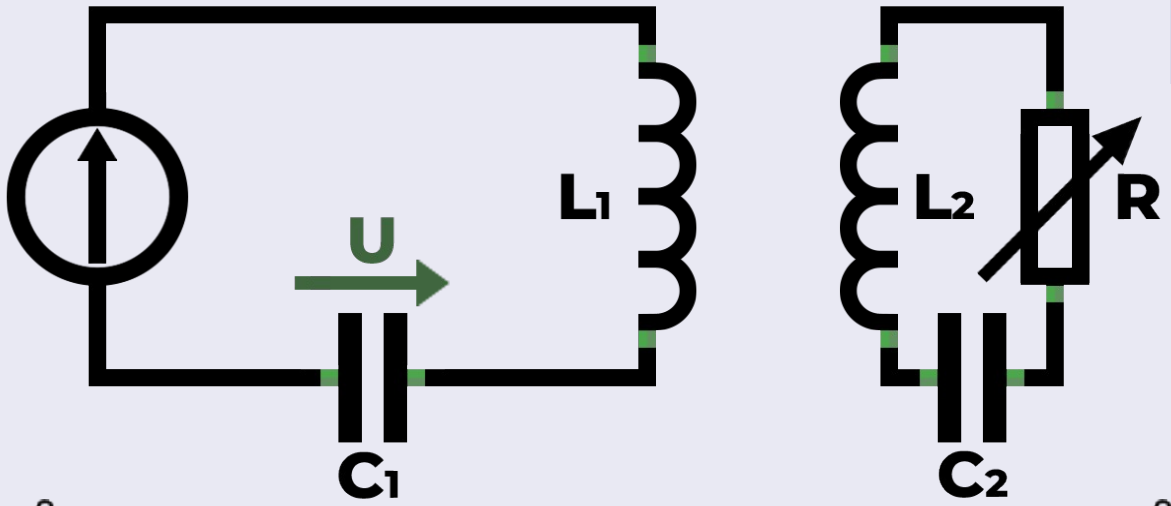
Carte :

- L_2 = 9 mH
- C_2 = 0,22 µF
- R = 0 ou 1kΩ

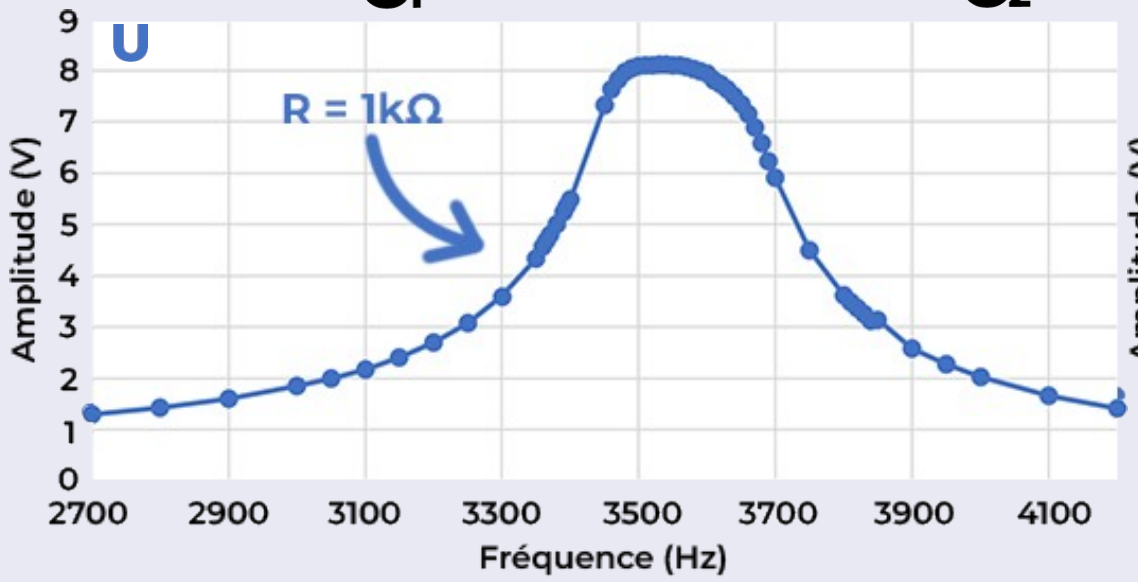
Fréquence de résonance de la carte et du lecteur : **3500 Hz**

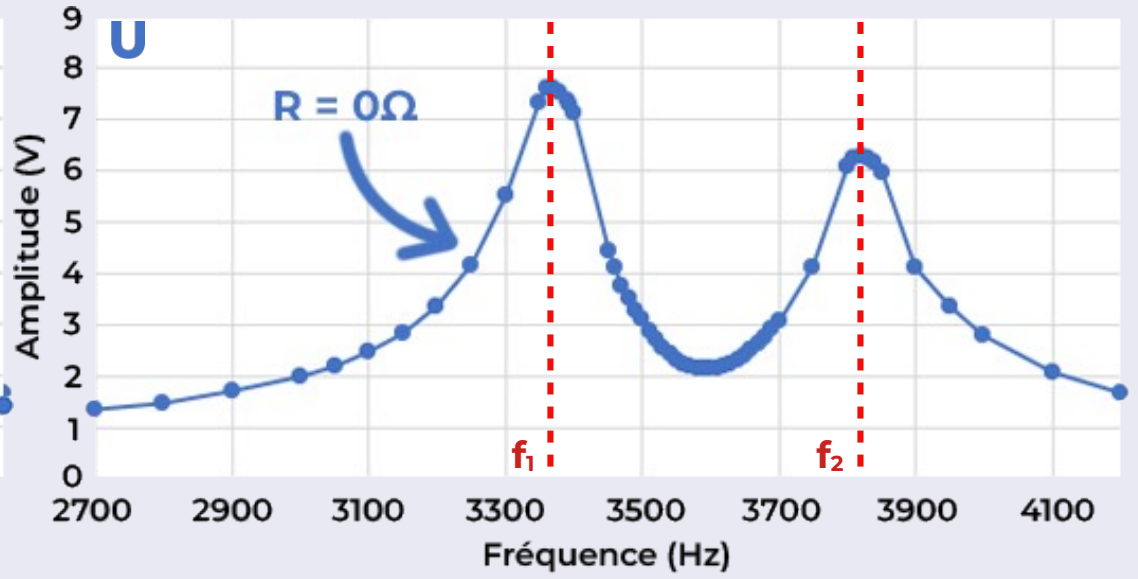
II / Modélisation de la liaison carte-lecteur

Fréquences de résonnance :



Équations de couplage :	Fréquences
$u_e = u_{C1} + L_1 \frac{di_1}{dt} + M \frac{di_2}{dt}$	$f_1 = \frac{f_0}{\sqrt{1+k}}$
$0 = u_{C2} + u_R + L_2 \frac{di_2}{dt} + M \frac{di_1}{dt}$	$f_2 = \frac{f_0}{\sqrt{1-k}}$





II / Modélisation de la liaison carte-lecteur

Optimisation du montage :

Bande passante :

$$A(f_{C1}) = A(f_{C2}) = \frac{A_{max}}{\sqrt{2}}$$

Résultats :

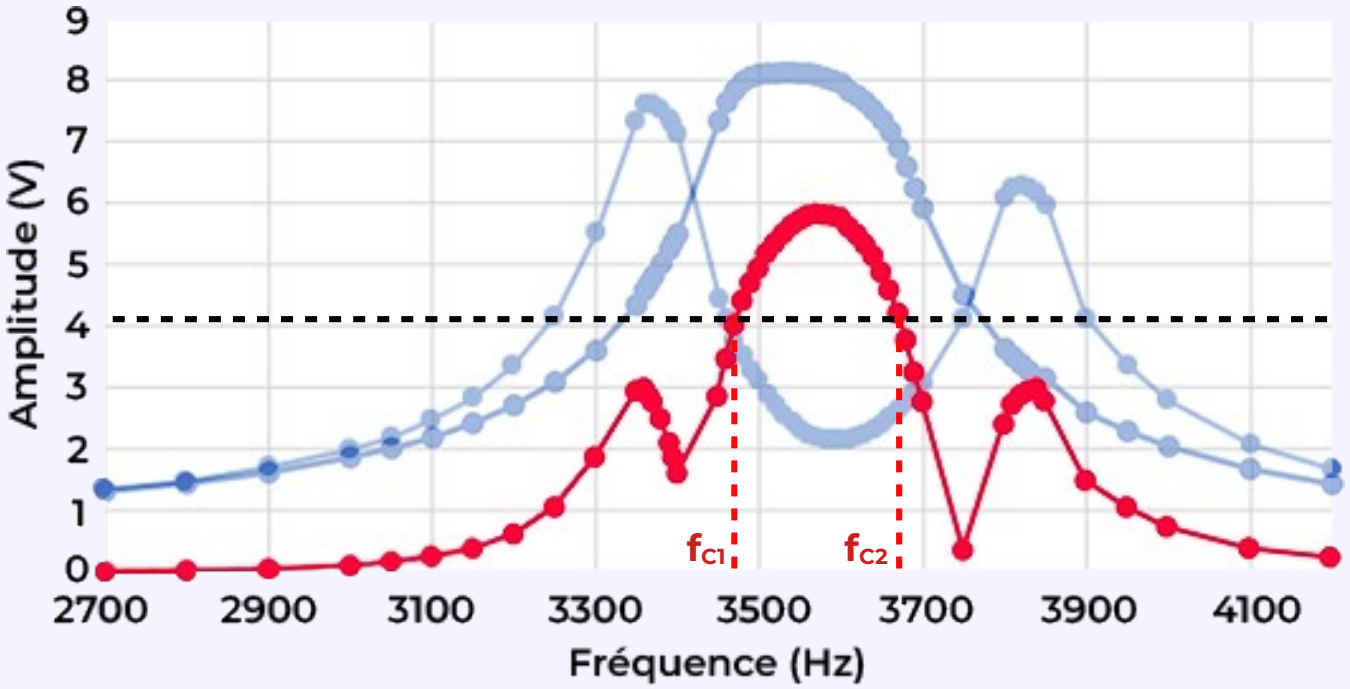
f_{C1} = 3470 Hz

f_{C2} = 3670 Hz

Δf = 200 Hz

EXPÉRIENCE : choix de la fréquence

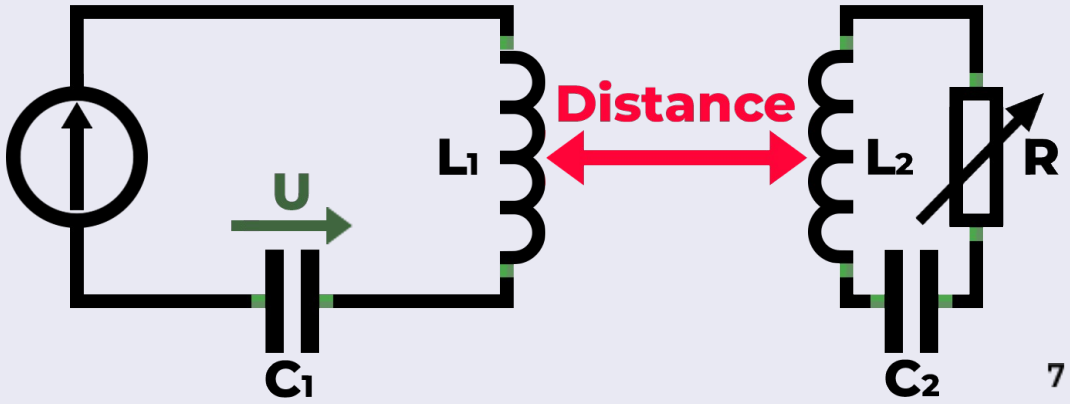
Écart de tension aux bornes du condensateur du lecteur



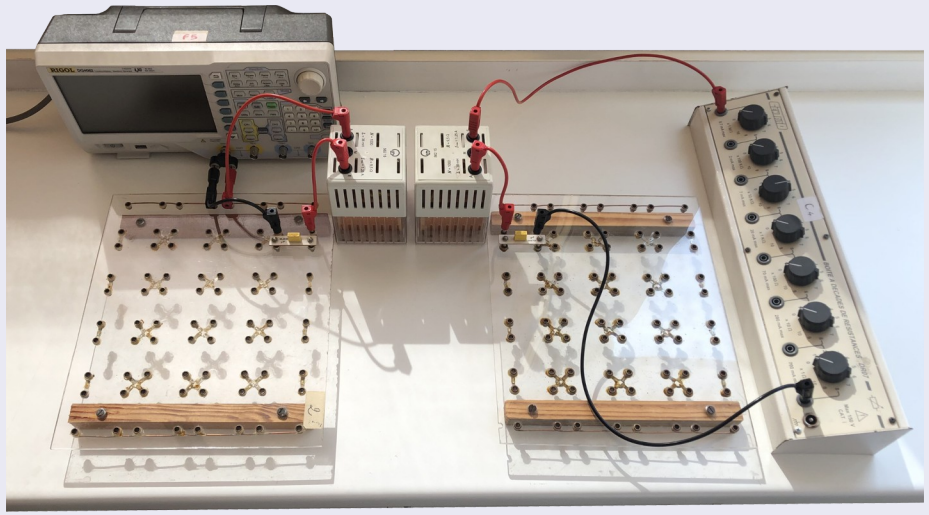
écart maximal entre les 2 courbes pour f = 3540 Hz

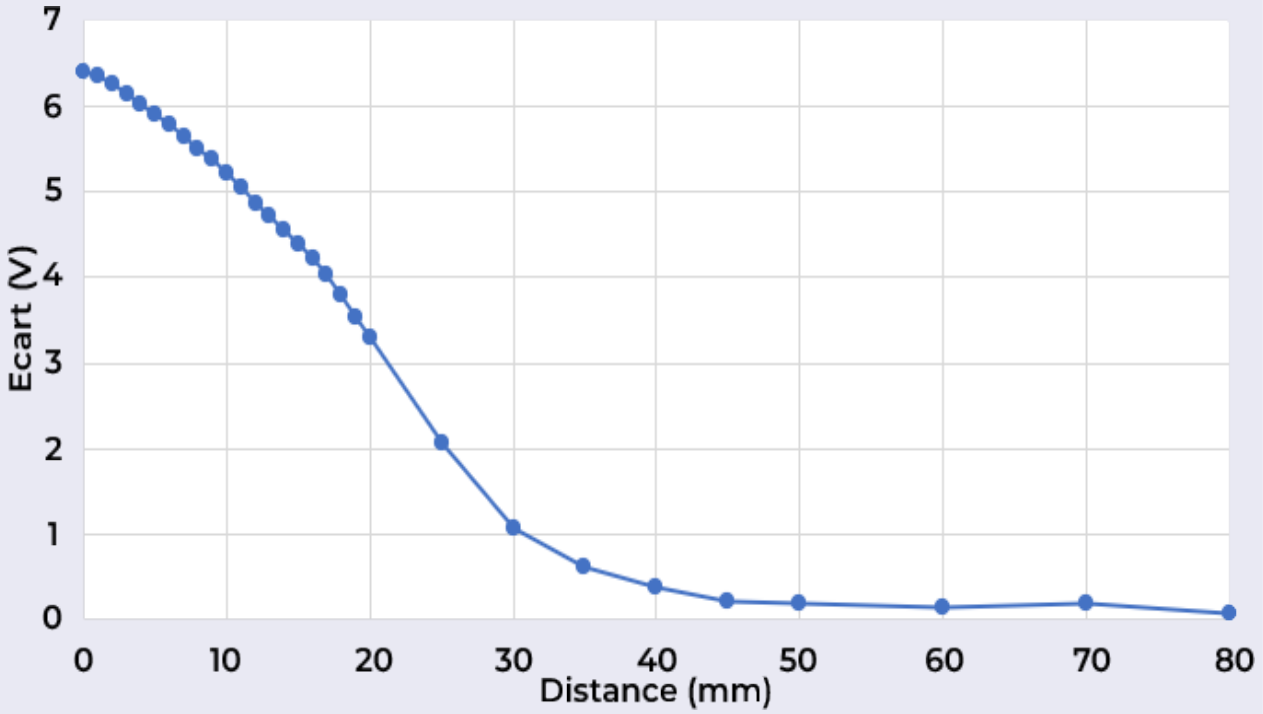
II / Modélisation de la liaison carte-lecteur

EXPÉRIENCE : Écart en fonction de la distance



Écart = $U_{R=0\Omega} - U_{R=1k\Omega}$

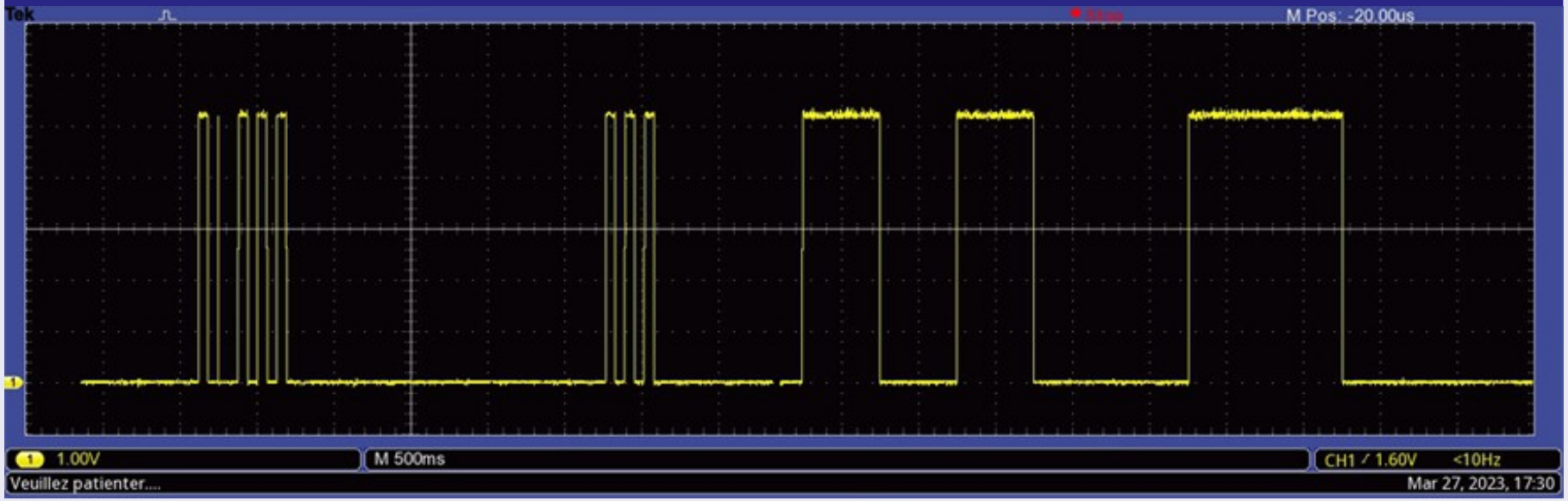




III / Transmission de notre propre message

Envoi du message :

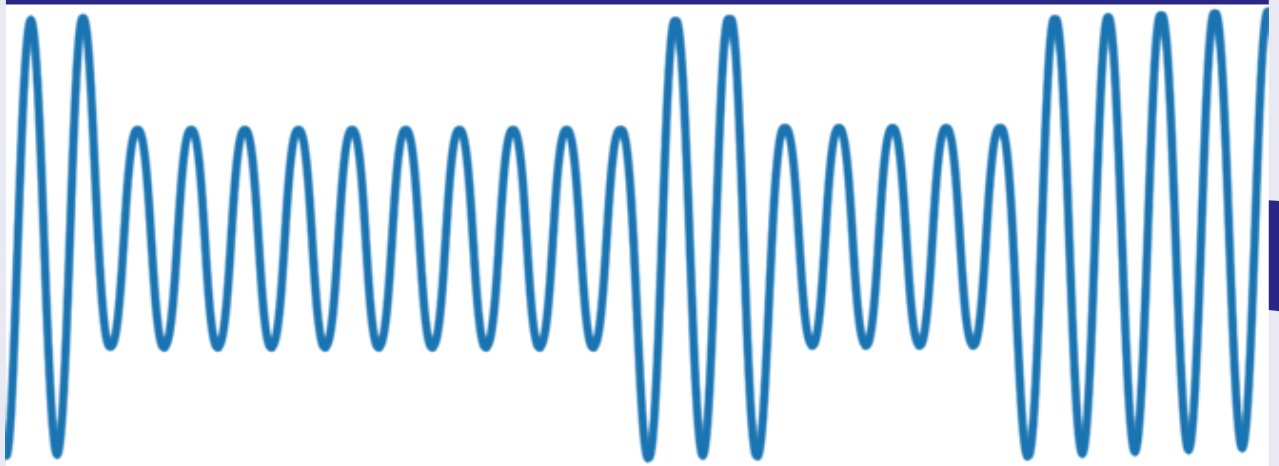
Résultat :



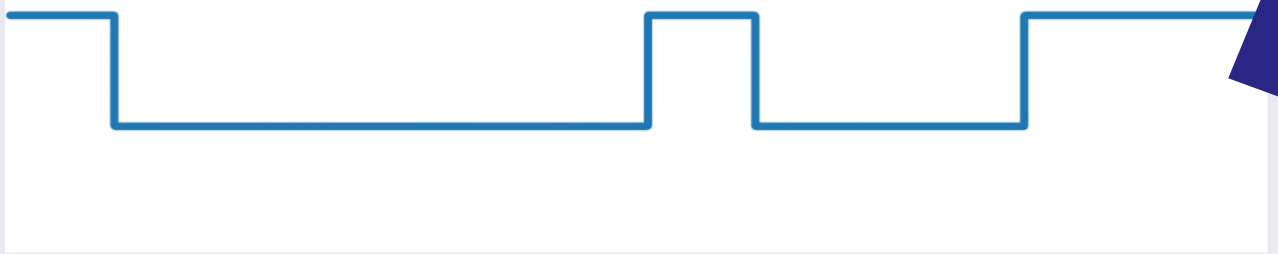
III / Transmission de notre propre message

Réception du message :

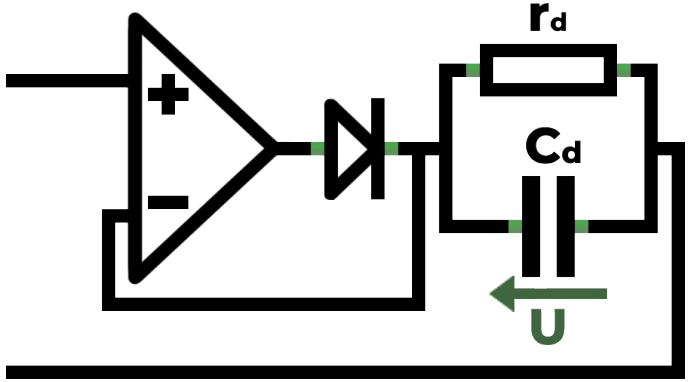
Signal observé dans le lecteur :

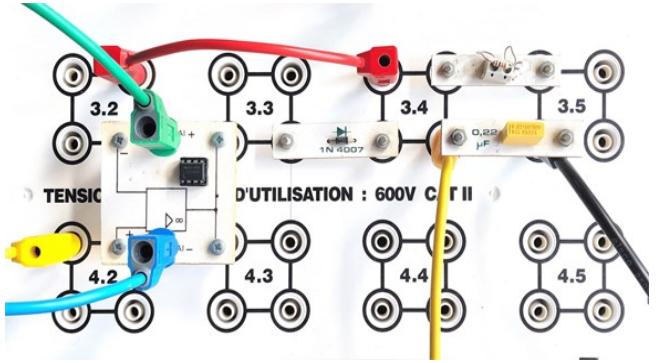


Signal porteur de l'information :



Démodulation :

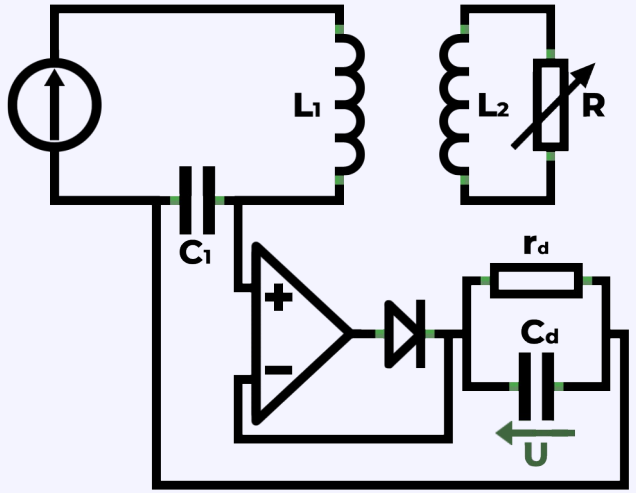


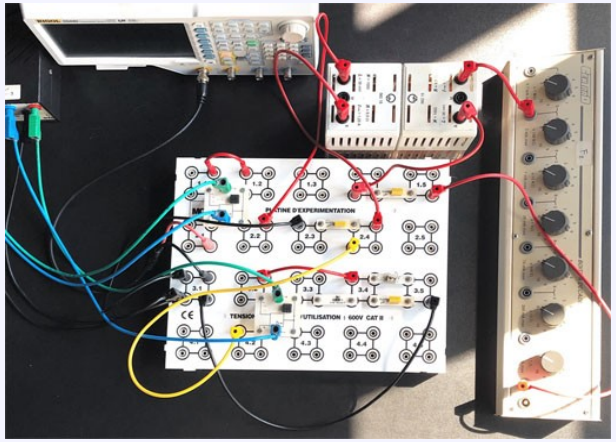


II / Modélisation de la liaison carte-lecteur

Démodulation d'amplitude :

Dispositif :

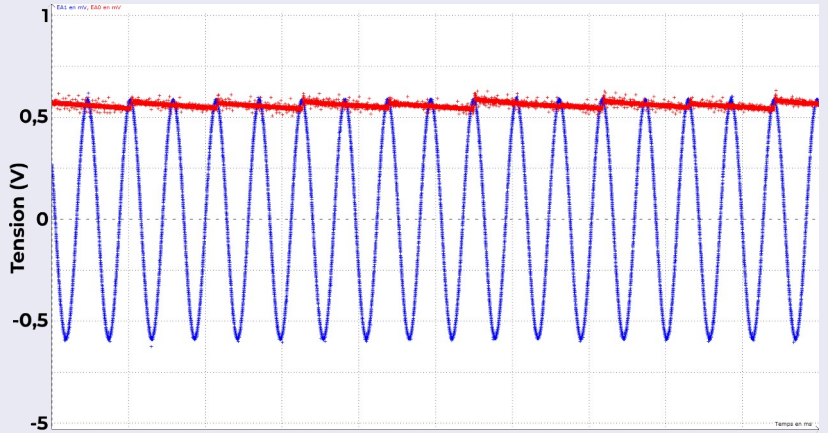


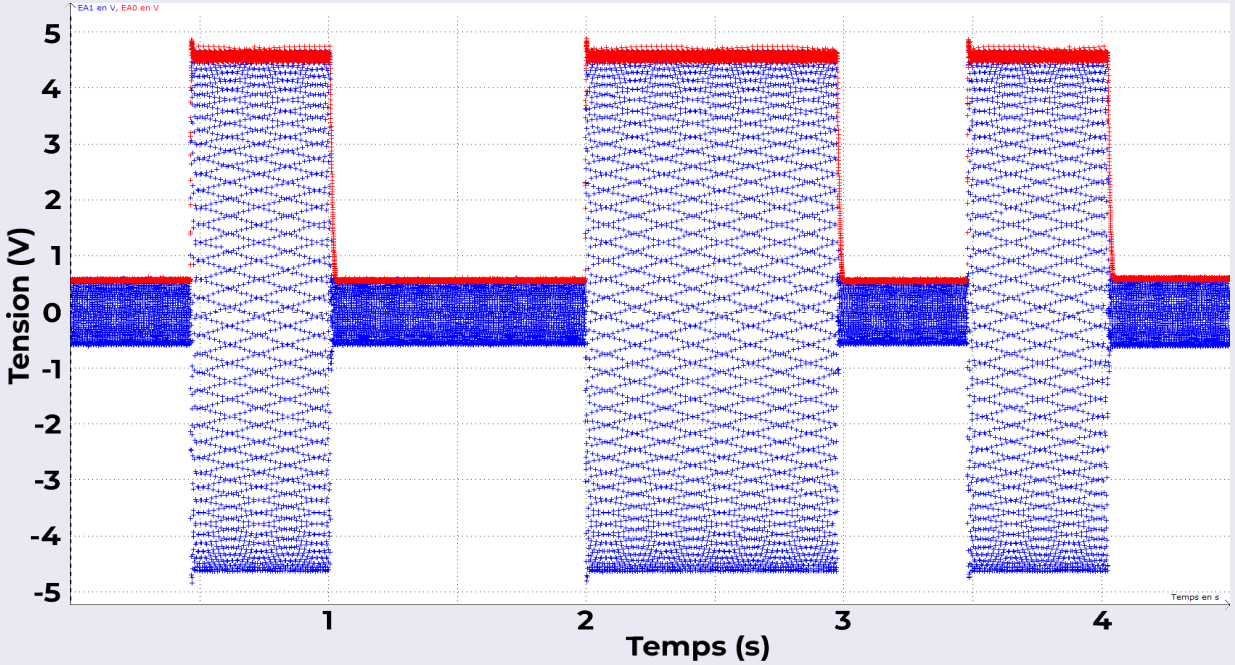


Caractéristiques :

$r_d = 10k\Omega$

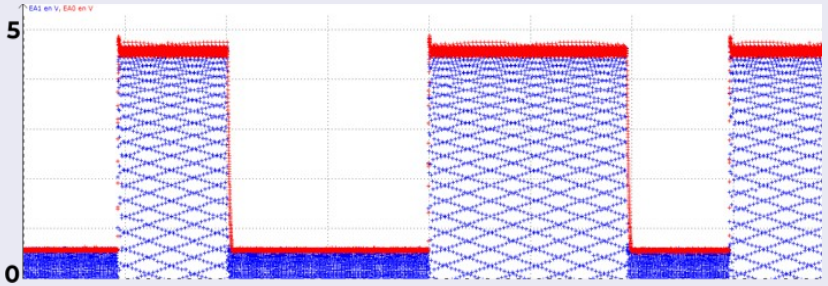
$C_d = 2{,}4\ \mu F$

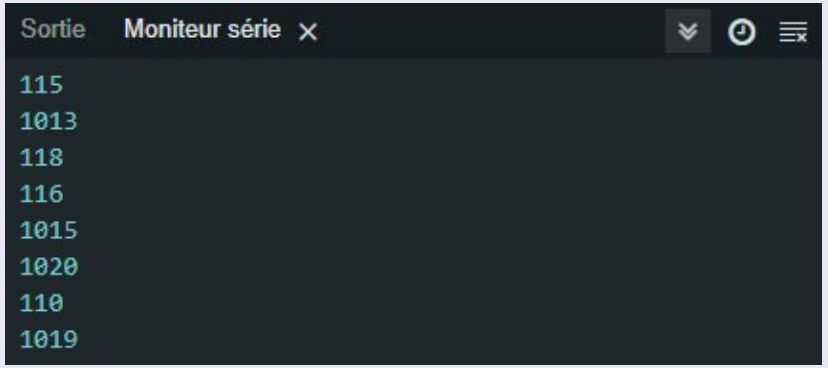


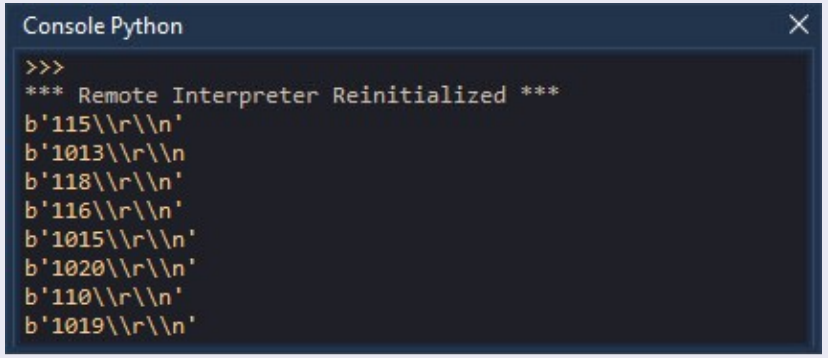


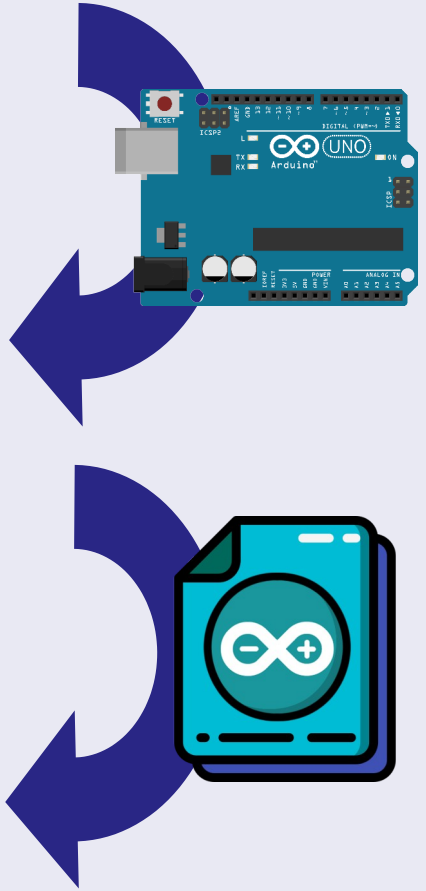
III / Transmission de notre propre message

Réception du message :

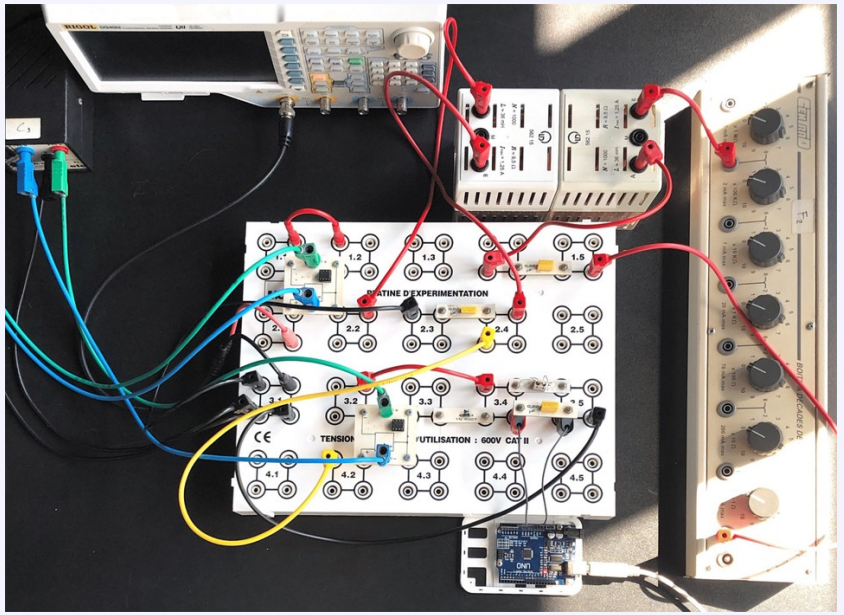








Arduino :



```
// Boucle d'acquisition
void loop() {
current_tension = analogRead(tension);
Serial.println(current_tension);
delay(500);
}
```

III / Transmission de notre propre message

Réception du message :

Console Python

```
rawdata =  ["b'115\\r\\n'", "b'1013\\r\\n'",
"b'118\\r\\n'", "b'116\\r\\n'", "b'1015\\r\\n'",
"b'1020\\r\\n'", "b'110\\r\\n'", "b'1019\\r\\n'"]
>>> nettoie(rawdata)
[115, 1013, 118, 116, 1015, 1020, 110, 1019]
```

Nettoie

Console Python

```
cleandata = [115, 1013, 118, 116, 1015, 1020, 110, 1019]
>>> compare(cleandata)
[0, 1, 0, 0, 1, 1, 0, 1]
```

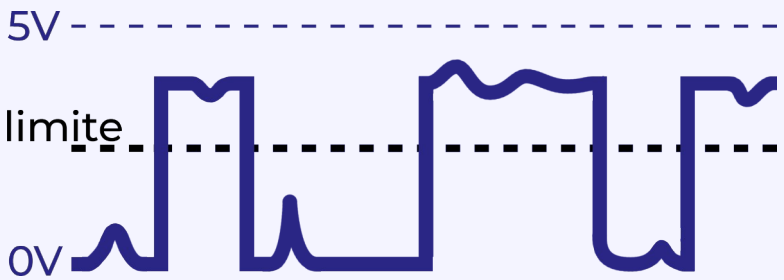
Compare

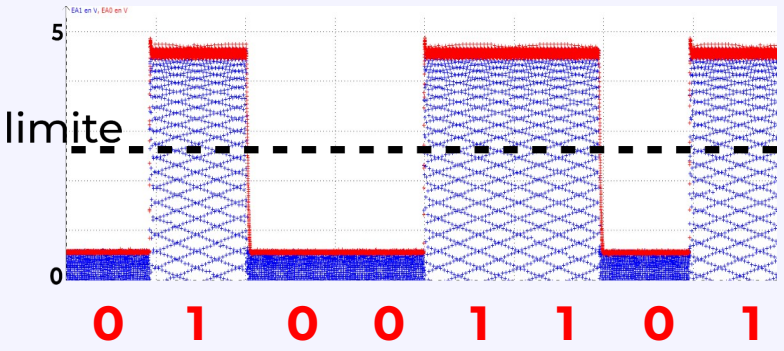
Console Python

```
binairedata =  [0, 1, 0, 0, 1, 1, 0, 1]
>>> traduit(binairedata)
'M'
```

Traduit

Comparaison :





CONCLUSION

Comment dialoguer et récupérer les informations contenues sur une carte de transport ?

Annexes

code.ino

```
1   const int tension = A0;
2
3   int current_tension;
4
5   // Fonction d'initialisation
6   void setup() {
7   Serial.begin(9600);
8   pinMode(13, OUTPUT);
9   }
10
11  // Boucle d'acquisition
12  void loop() {
13  current_tension = analogRead(tension);
14  Serial.println(current_tension);
15  delay(500);
16  }
```

Annexes

```
""" Importation des modules """
import serial                    # module de communication arduino/python
import matplotlib.pyplot as plt # module pour tracer les résultats


"""Ouverture de la liaison série"""
port = "COM6"      # COM35 est le port utilisé sur notre arduino

try : arduino = serial.Serial(port,timeout=1)
except: print('Vérifier le port utilisé')


"""Initialisation des variables"""
tension_max = 5     # valeur max en Volt enregistrée par l'arduino
tension_mid = 3.5   # valeur en Volt de la limite

limit = int(tension_mid * 1023/tension_max) # transformation de mid_tension en val entre 0 et 1023

nb_acq = 40 #nombre de valeurs enregistrées
nb_val = 8 #nombre de bits devant être traités

rawdata = []        # données brutes
compt = 0


"""Réception des données"""
arduino.readline()  # on écarte la première donnée qui est vide dans notre cas

while compt < nb_acq :
    rawdata.append(str(arduino.readline()))
    compt+=1

arduino.close()
```

Annexes

```
"""Nettoyage et Stockage des données"""

def nettoie(L:list)->list :
    '''
    arg : L = ["b'125;y\\r\\n'", ...]
    ret : [125, ...]
    '''
    cleanL = []
    for elt in L :
        temp = elt[2:-5]            # "b'xxx;y\\r\\n'" -> "xxx;yy"
        cleanL.append(int(temp))    # ["xxx","yy"] -> [xxx,yy]
    return cleanL

cleaned_data = nettoie(rawdata)
print('cleaned_data : ',cleaned_data)

def write(L:list):
    '''inscrit les données de L dans data.txt
    '''
    compt=0
    file = open("data.txt",mode="w")
    for elt in L :
        compt+=1
        temp = str(elt)
        file.write(temp+';'+str(compt)+'\n')
    file.close()


write(cleandata)    # inscription des valeurs dans le dossier data.txt

vals,temps = np.loadtxt("data.txt",delimiter=";",unpack=True)
```

Annexes

```
"""Traitement des données"""

def reception(L:list,n:int)->list:
    '''élimine le bit signal et renvoie les n bits qui le suivent'''
    i=0
    while L[i]==0 and i<len(L):  #élimination des premiers 0
        i+=1
    i+=1            #élimination du bit de signal (premier 1)
    if i+n>len(L):
        print('le message est coupé, veuillez réessayer')
    else :
        return L[i:i+n]

recep_data = reception(cleaned_data,nb_val)
assert type(recep_data) == list



def compare(L:list)->list:
    """transforme les tensions (valeurs entre 0 et 1023) enregistrées dans L
    et renvoie les valeurs binaires correspondantes"""
    newL=[]
    for elt in L :
        if elt > limit :
            newL.append(1)
        else :
            newL.append(0)
    return newL

binaire_data = Tension_vers_Binaire(recep_data)
print('binaire_data : ',binaire_data)
```

Annexes

```
def traduit(L:list)->str:
    """transforme une chaine binaire renvoie le texte correspondant"""
    text = "" # chaîne vide qui va contenir le texte final
    octets = [L[i:i+8] for i in range(0, len(L), 8)] # Sépare la liste en chunks de 8bits

    for octet in octets:
        ascii_val = 0

        for i in range (8):
            ascii_val += octet[7-i]*(2**i) #calcul de la valeur du chunk en base 10

        char = chr(ascii_val) # Transforme le code ascii en caractère
        text += char
    return text


message = Binaire_vers_Texte(binaire_data)
print('message : ',message)


write(binaire_data)    # inscription des valeurs dans le dossier data.txt

vals,temps = np.loadtxt("data.txt",delimiter=";",unpack=True)



"""Tracé des valeurs"""
plt.plot(temps,vals,"ob")
plt.show()
```