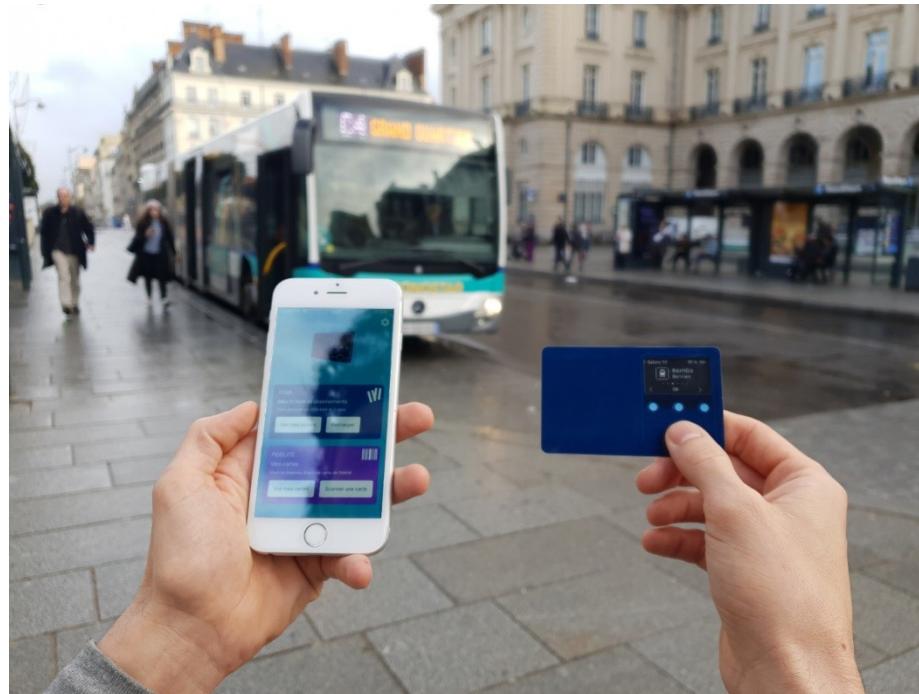
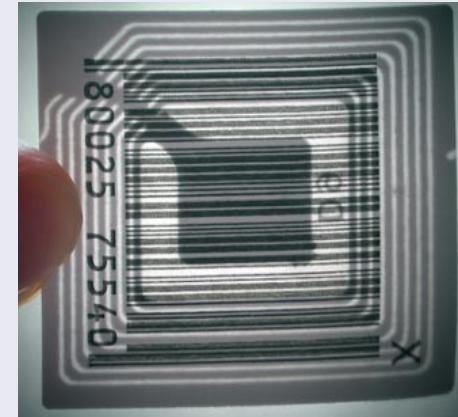


Communication des cartes à puce sans contact : principe et applications

Thème de l'année : la ville

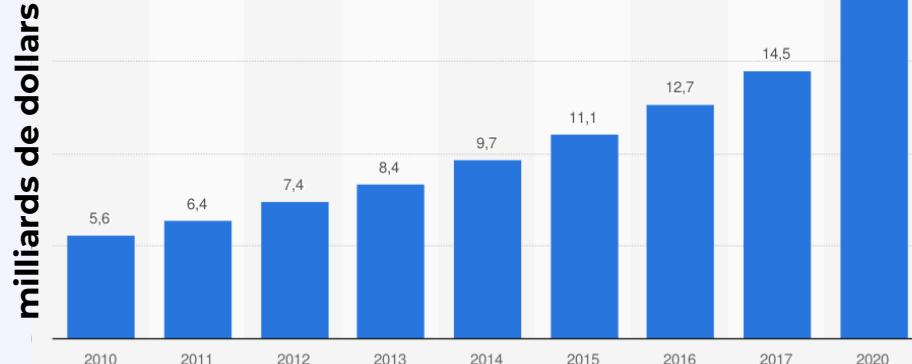


Applications :



Nouveaux dispositifs :

Marché de la RFID :



Problématique :

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

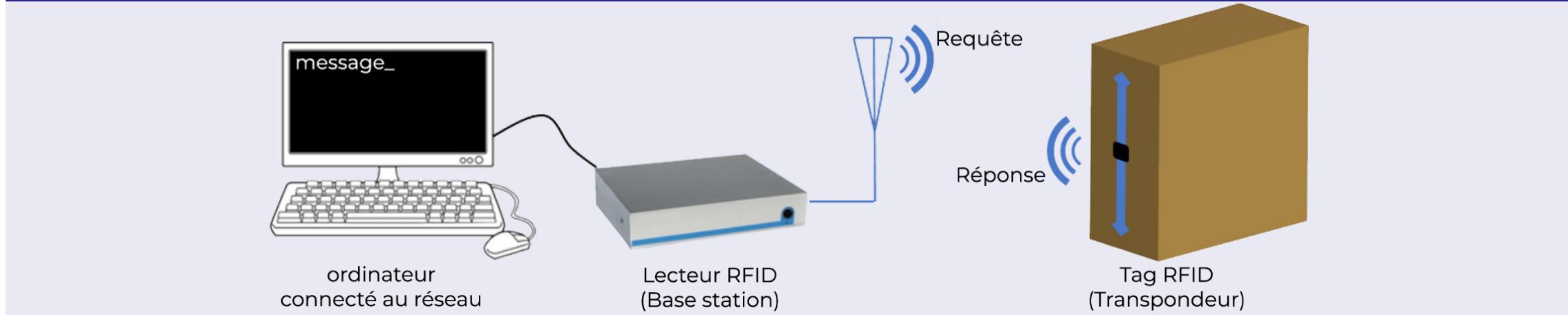
I – Communication par la technologie RFID

II – Modélisation de la liaison carte-lecteur

III – Transmission de notre propre message

I / Communication par la technologie RFID

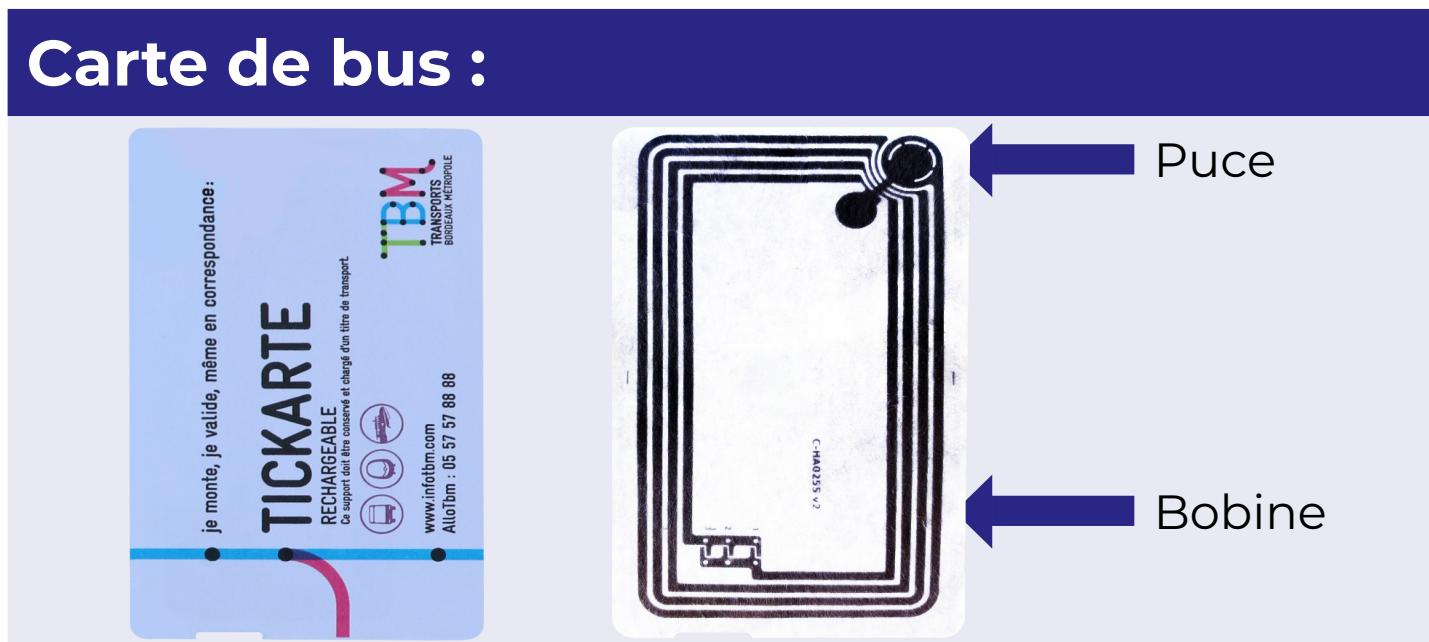
RFID : Radio Frequency Identification



Lecteur :

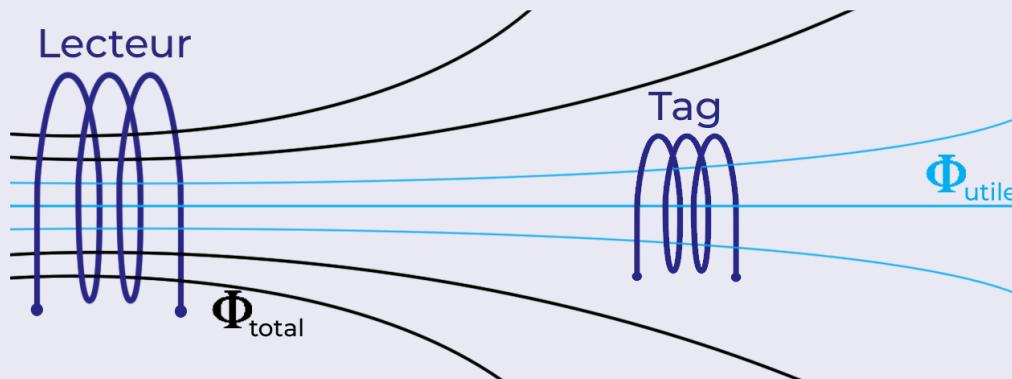


Carte de bus :



I / Communication par la technologie RFID

PRINCIPE : Induction magnétique



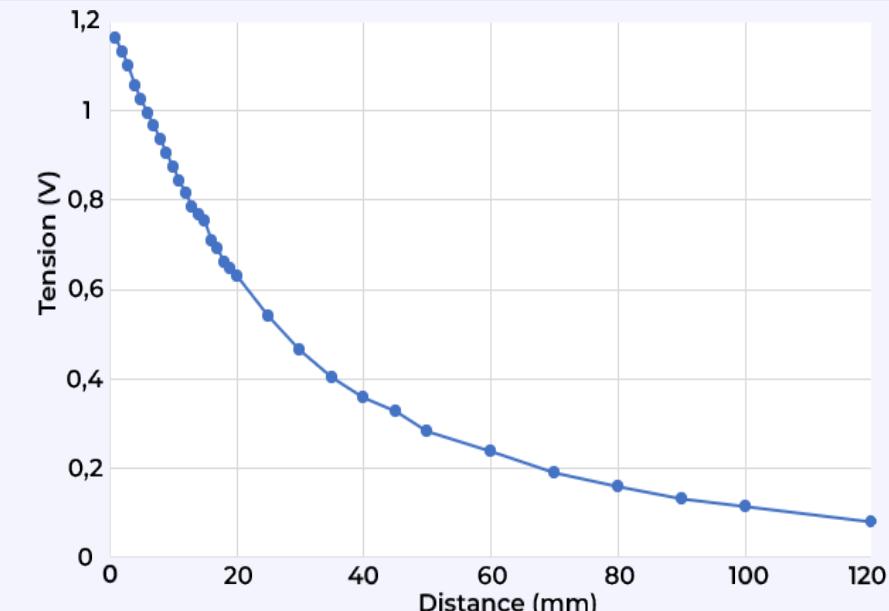
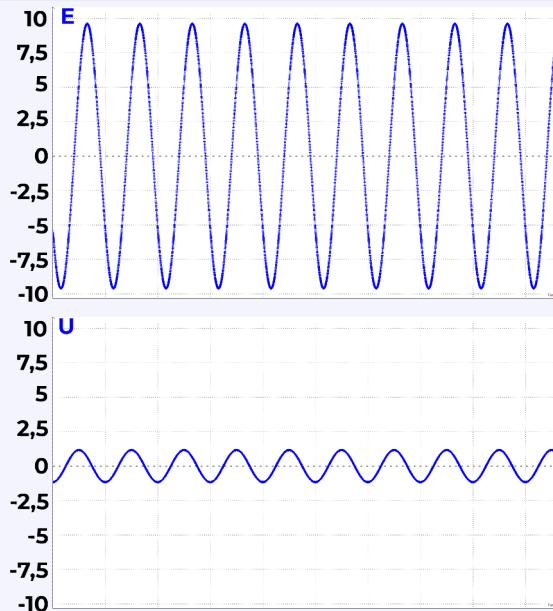
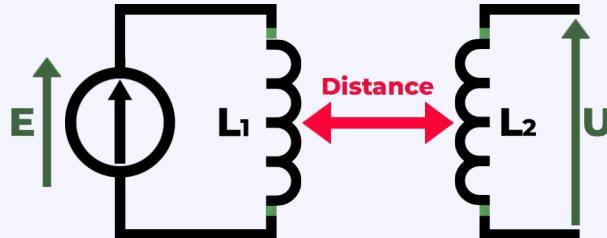
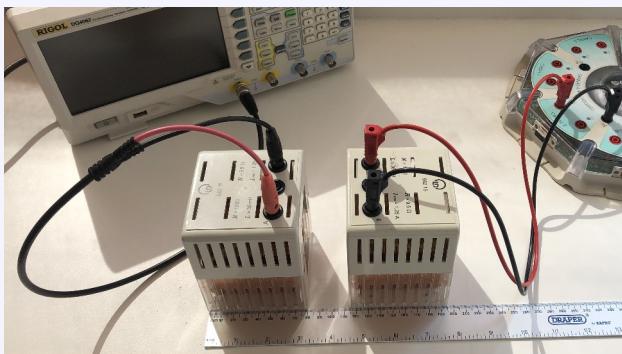
Loi de Faraday :

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

Flux :

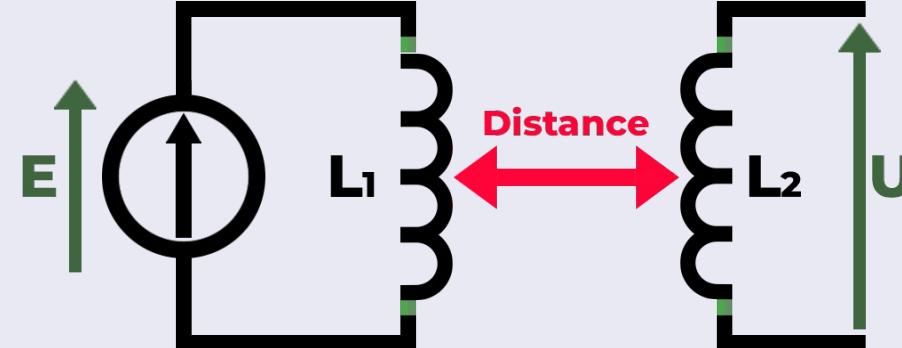
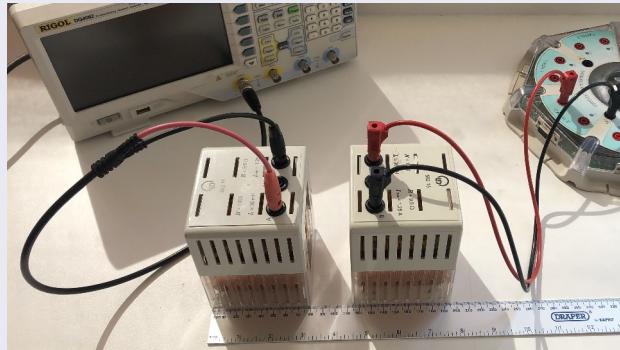
$$\Phi = \iint \vec{B} \cdot \vec{ds} = BNs$$

EXPÉRIENCE :



I / Communication par la technologie RFID

EXPÉRIENCE : Influence de la fréquence sur l'induction



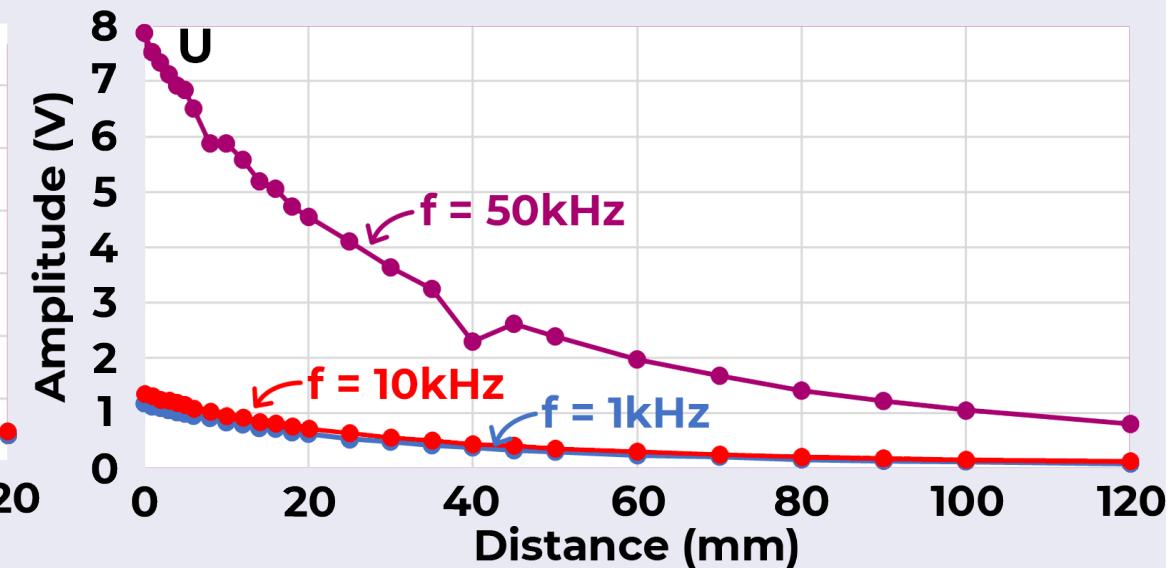
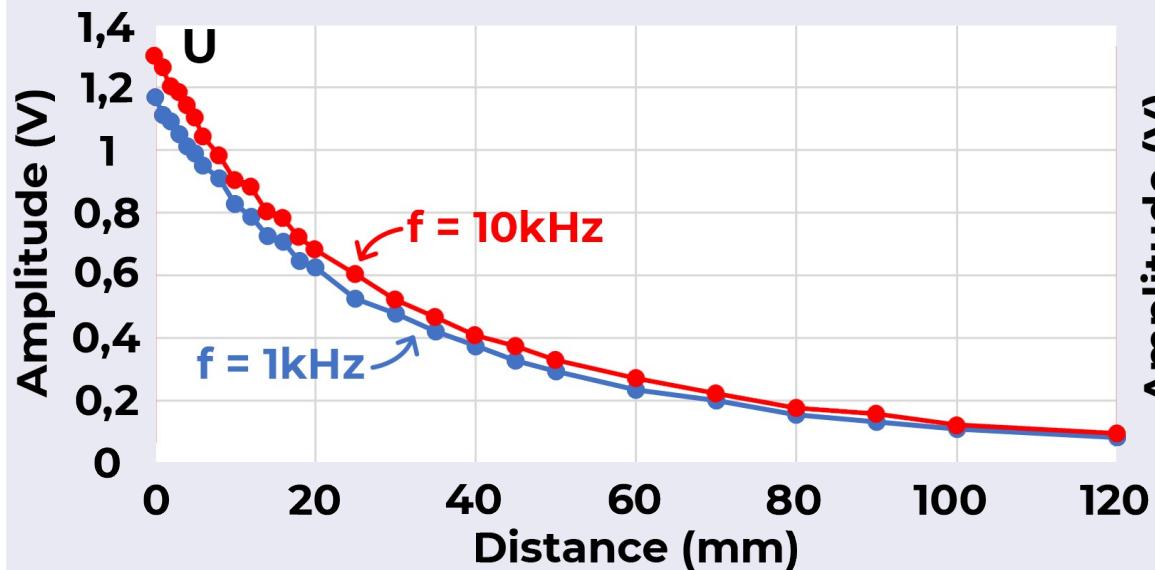
Signal :

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*



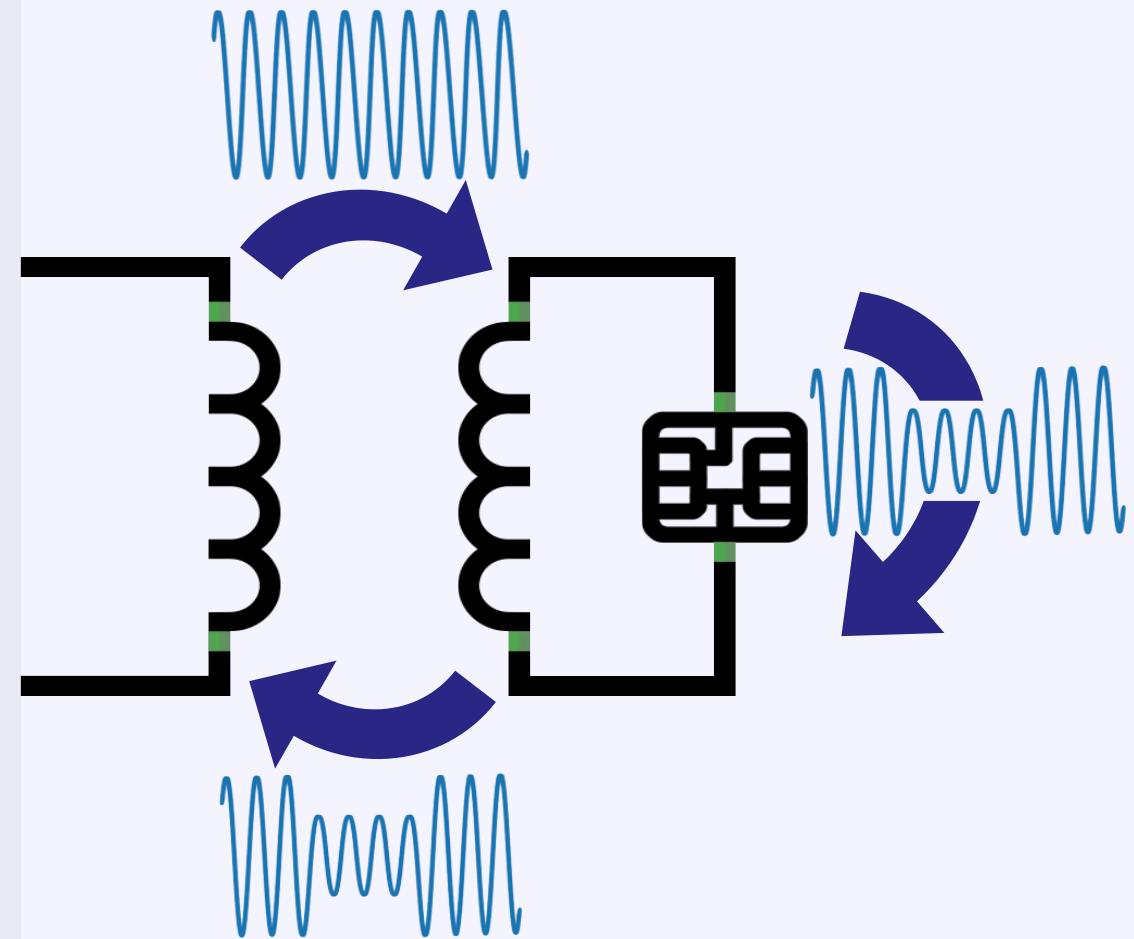
5,8 GHz : UHF

péages routiers



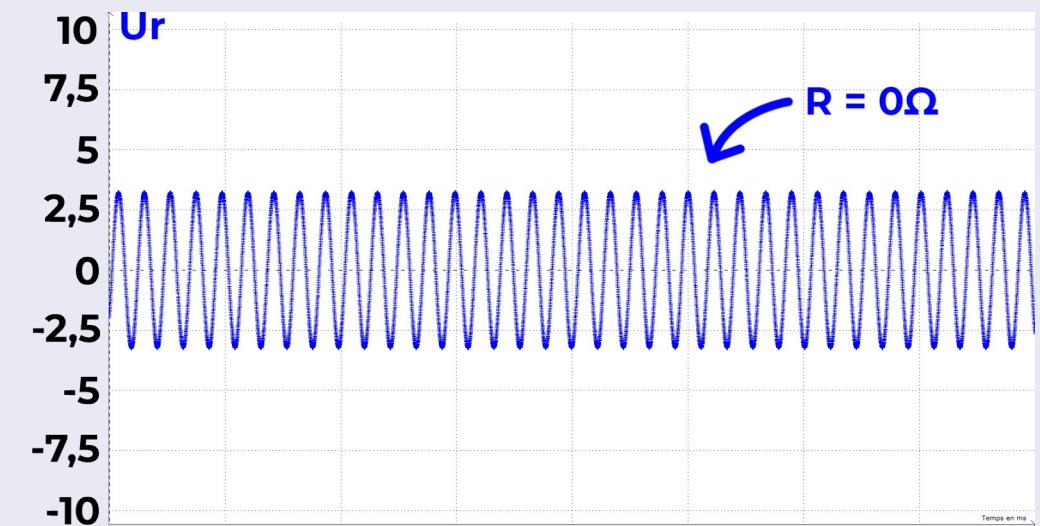
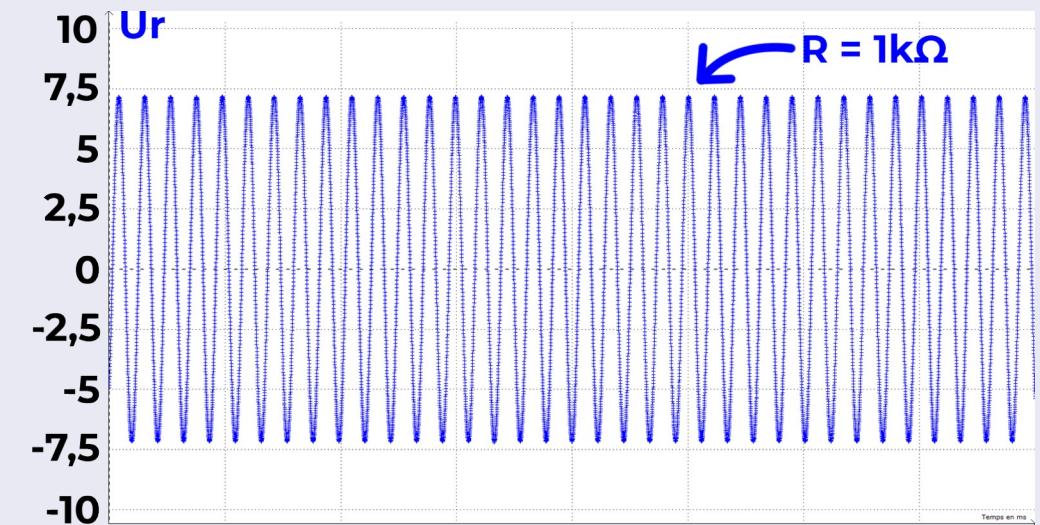
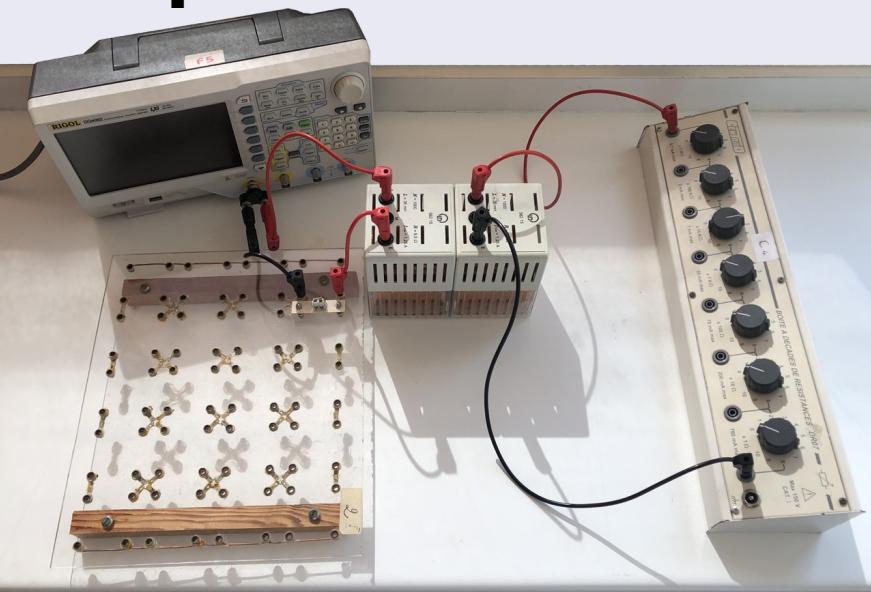
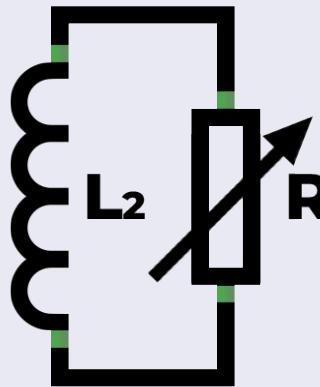
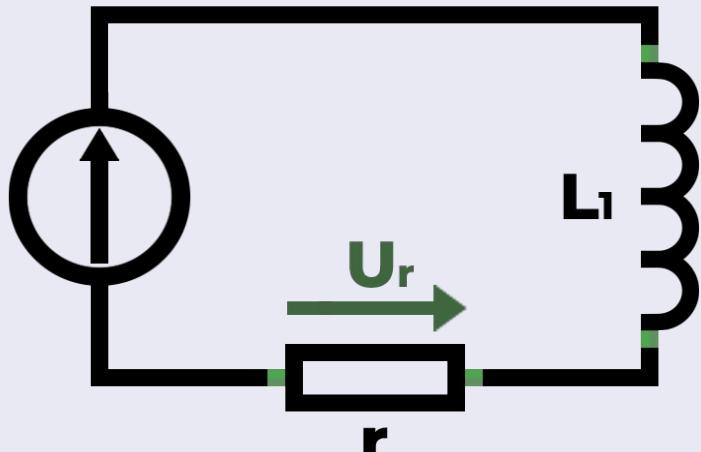
Couplage magnétique

Modulation :



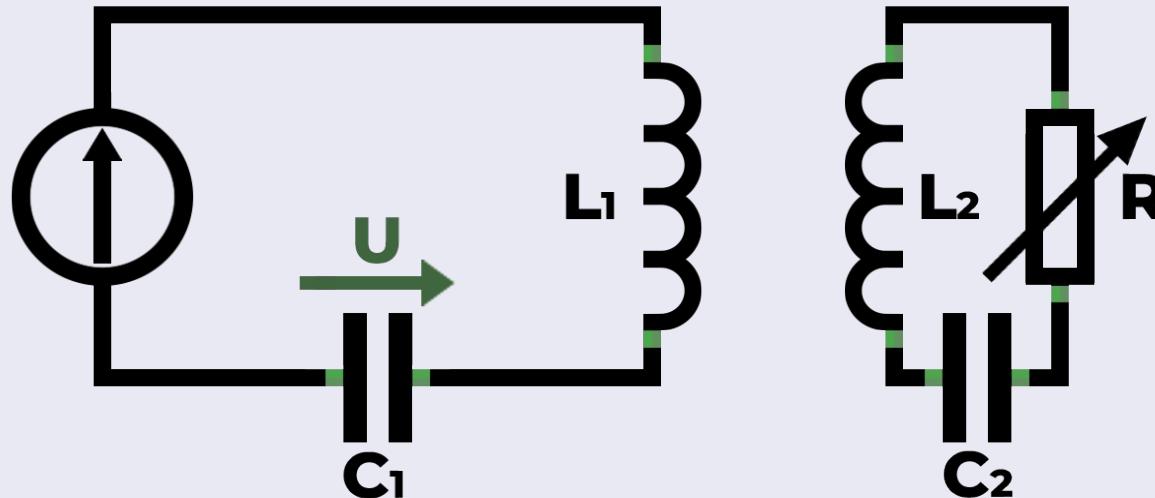
II / Modélisation de la liaison carte-lecteur

Modélisation initiale :



II / Modélisation de la liaison carte-lecteur

Modélisation finale :



Caractéristiques :

Lecteur :

- $L_1 = 9 \text{ mH}$
- $C_1 = 0,22 \mu\text{F}$

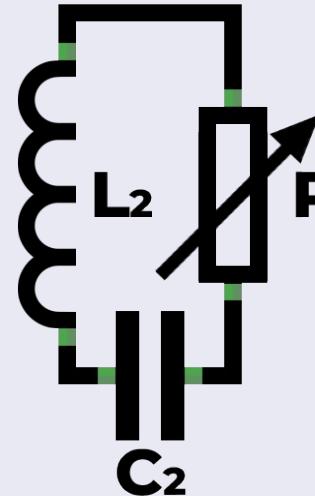
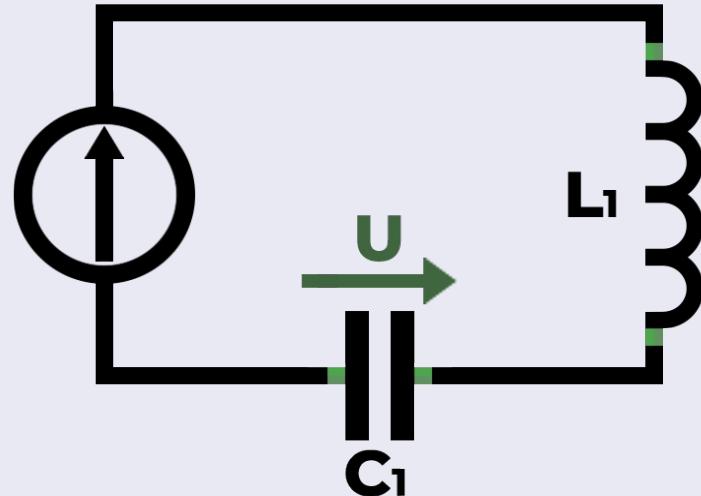
Carte :

- $L_2 = 9 \text{ mH}$
- $C_2 = 0,22 \mu\text{F}$
- $R = 0 \text{ ou } 1\text{k}\Omega$

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

II / Modélisation de la liaison carte-lecteur

Fréquences de résonnance :



Équations de couplage :

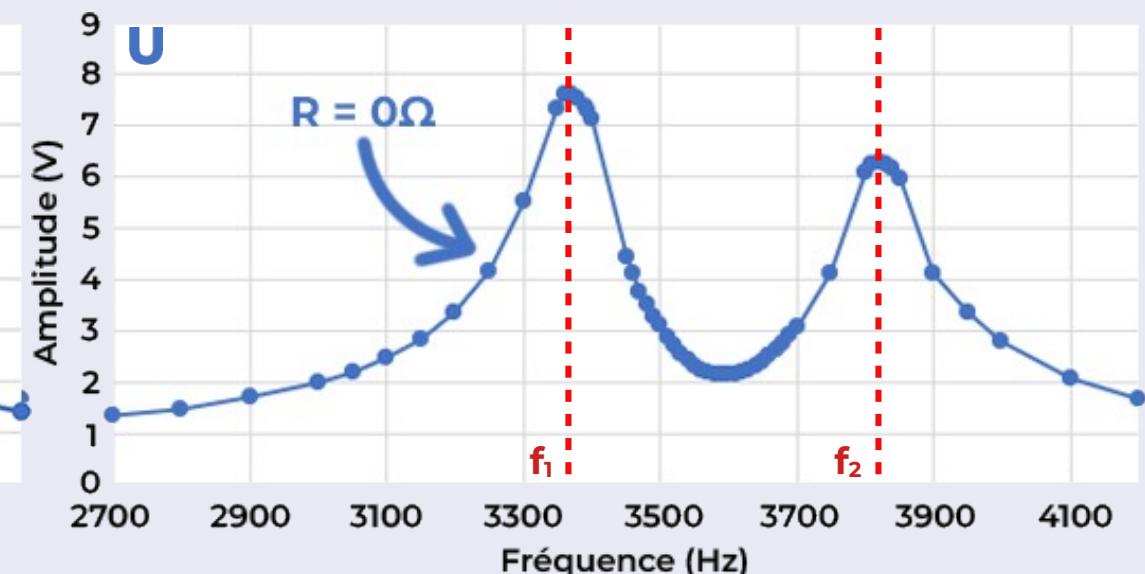
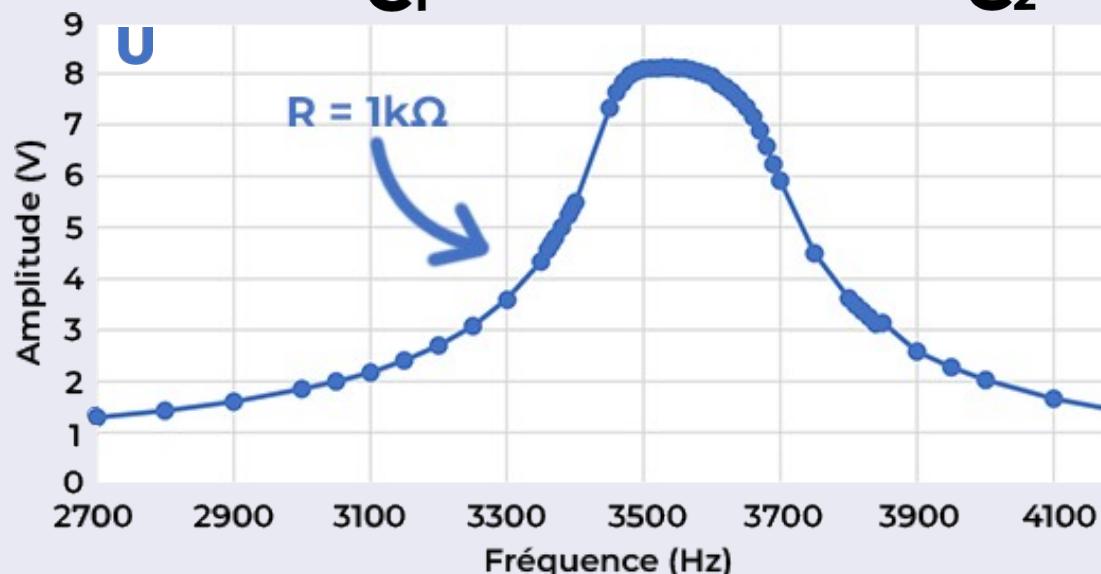
$$u_e = u_{C1} + L_1 \frac{di_1}{dt} + M \frac{di_2}{dt}$$

$$0 = u_{C2} + u_R + L_2 \frac{di_2}{dt} + M \frac{di_1}{dt}$$

Fréquences

$$f_1 = \frac{f_0}{\sqrt{1+k}}$$

$$f_2 = \frac{f_0}{\sqrt{1-k}}$$



II / Modélisation de la liaison carte-lecteur

Optimisation du montage :

Bandé passante :

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

Résultats :

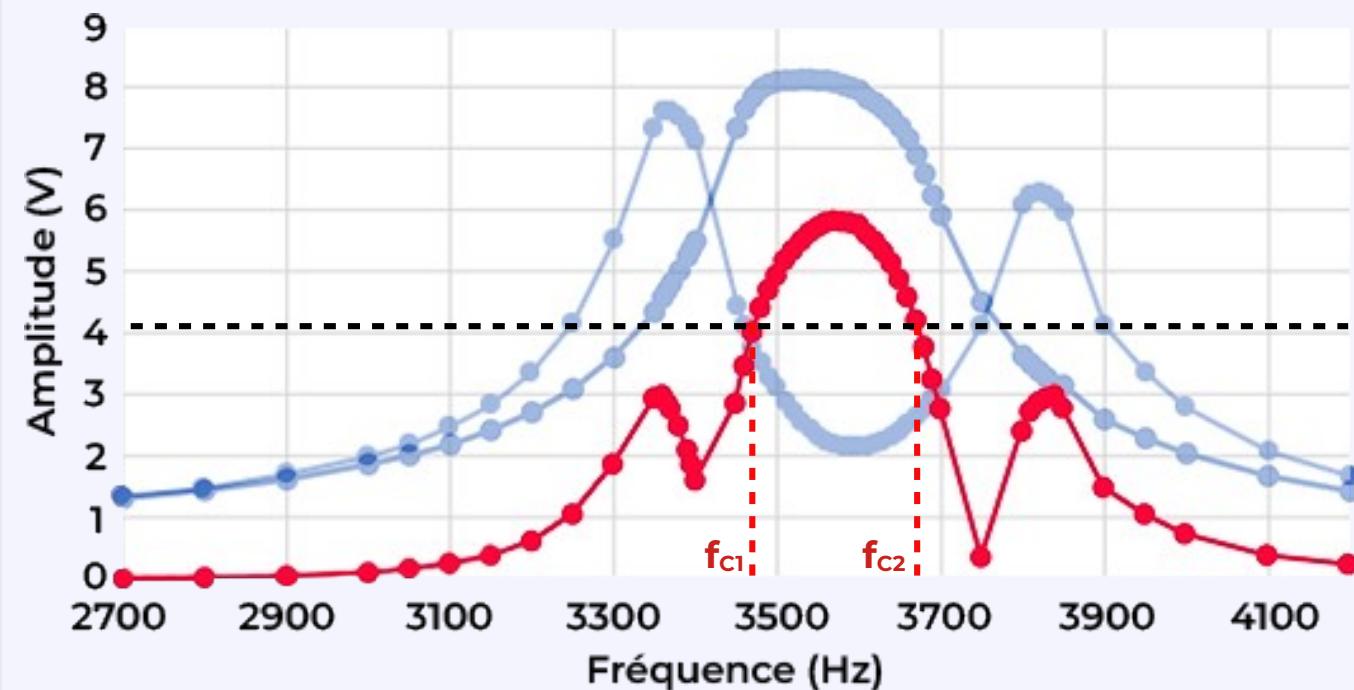
$$f_{C1} = 3470 \text{ Hz}$$

$$f_{C2} = 3670 \text{ Hz}$$

$$\Delta f = 200 \text{ Hz}$$

EXPÉRIENCE : choix de la fréquence

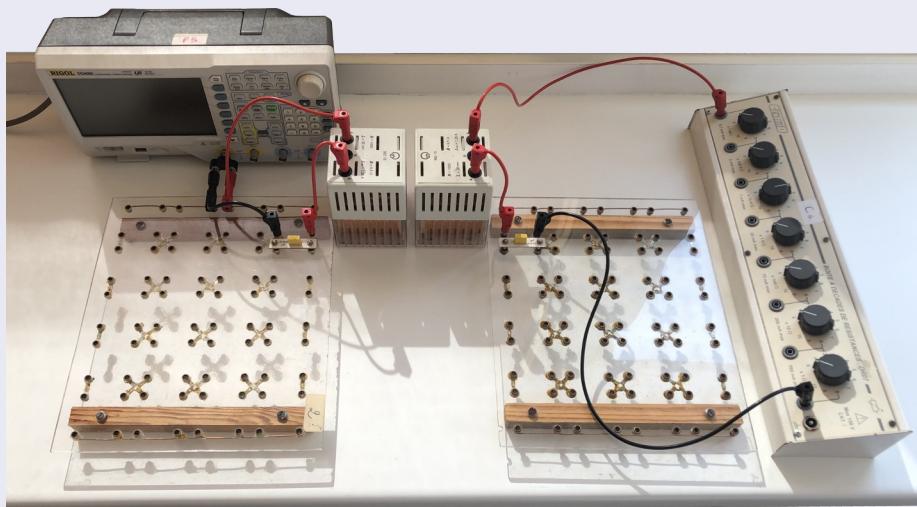
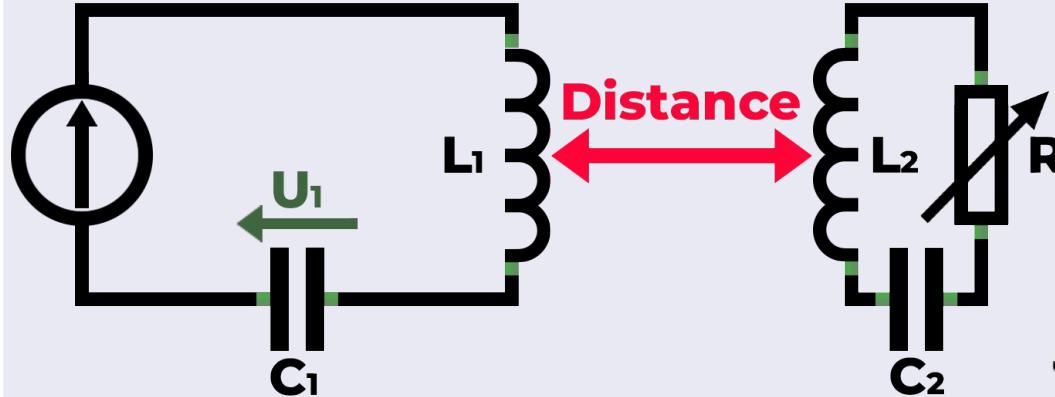
Écart de tension aux bornes du condensateur du lecteur



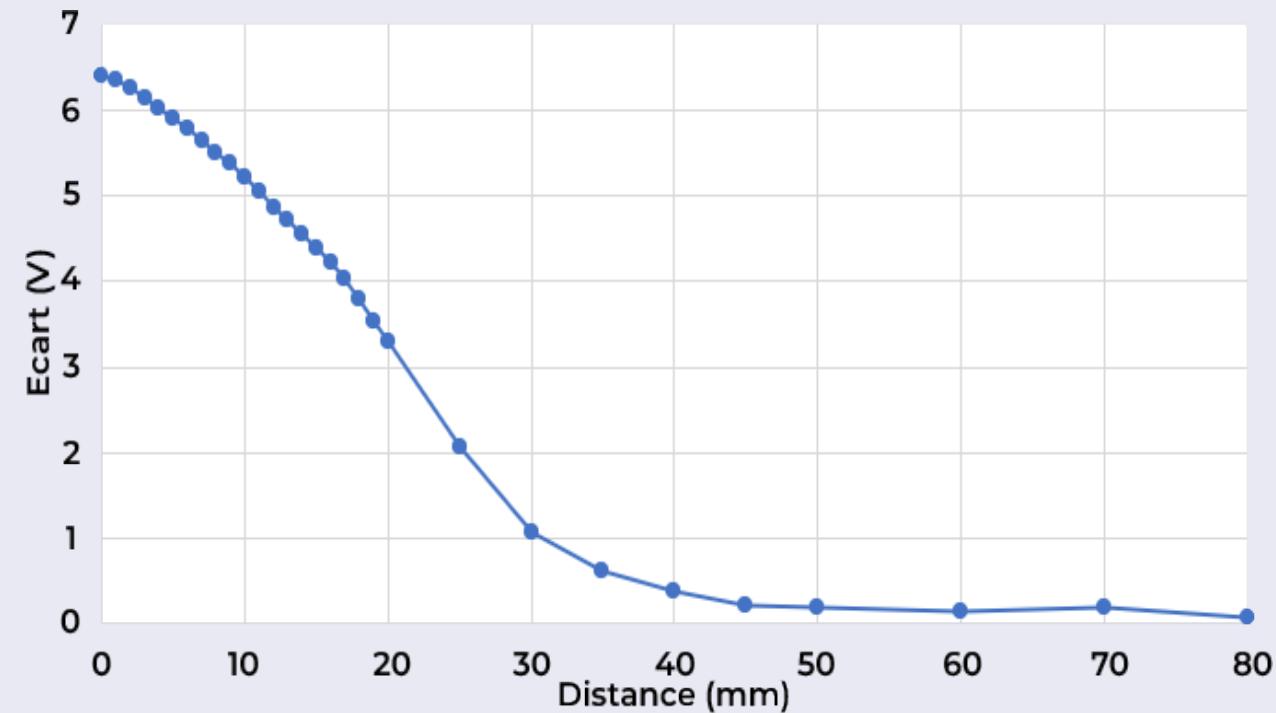
écart maximal entre les 2 courbes pour $f = 3540 \text{ Hz}$

II / Modélisation de la liaison carte-lecteur

EXPÉRIENCE : Écart en fonction de la distance

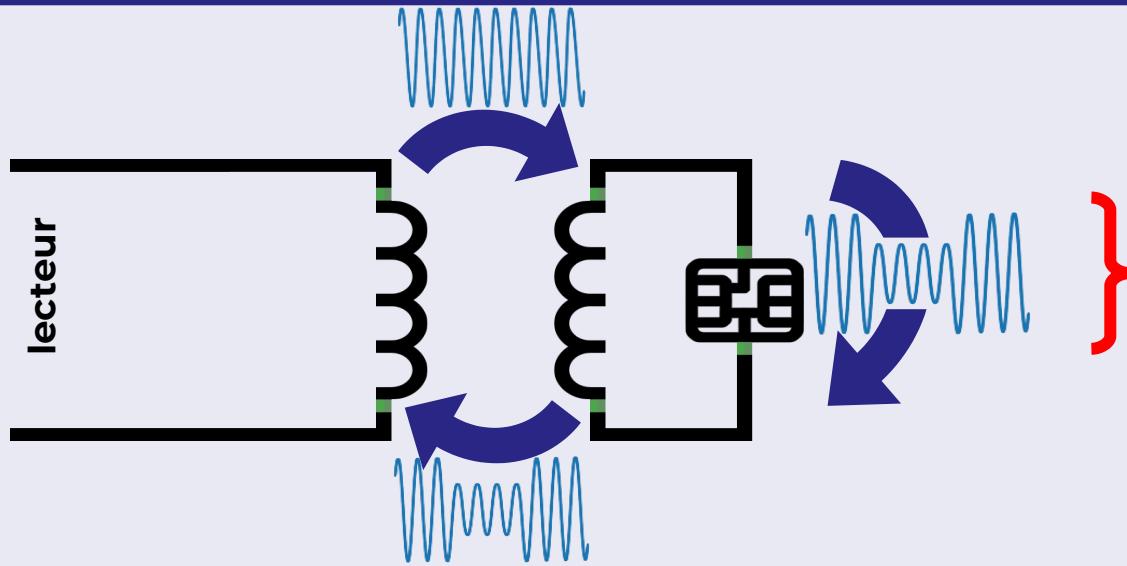


$$\text{Écart} = U_{R=0\Omega} - U_{R=1k\Omega}$$

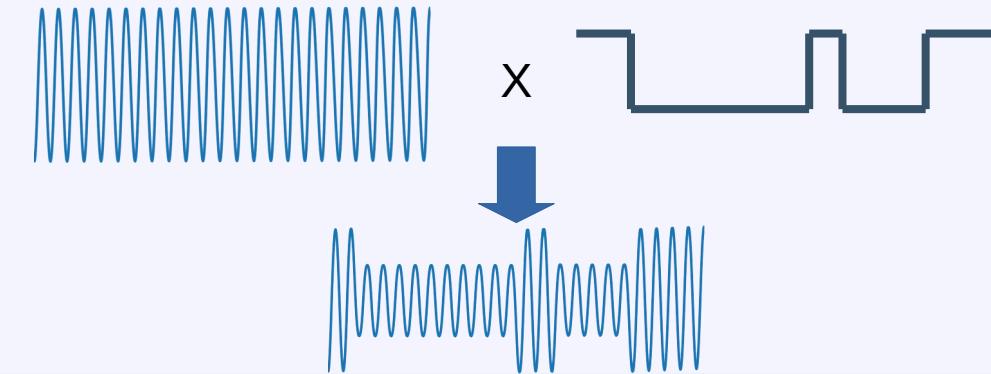


III / Transmission de notre propre message

Envoi du message :

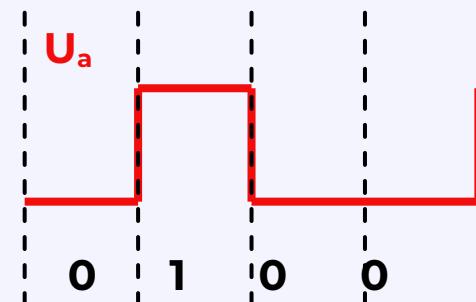
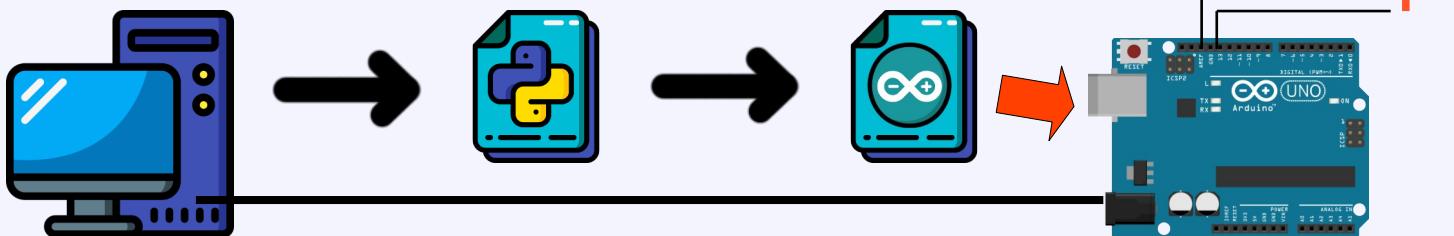


Réalisation :



Création signal créneau :

```
>>> texte_vers_binaire('MP')
'0100110101010000'
```



III / Transmission de notre propre message

Envoi du message :

Résultat :

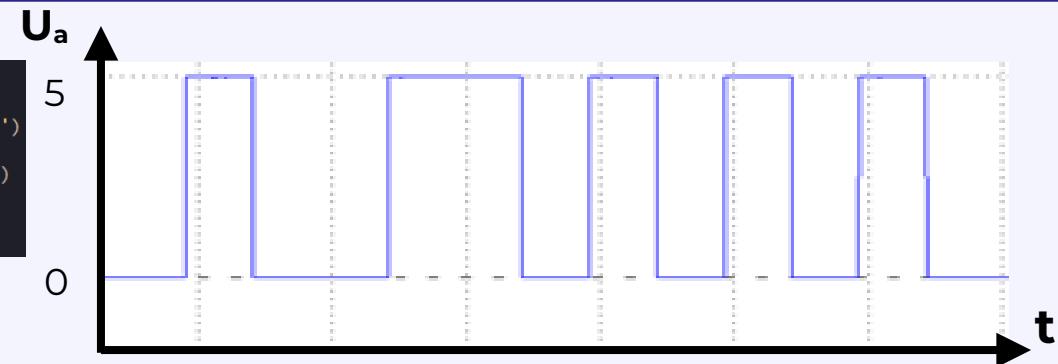


```
import serial
def texteversArduino(text):
    chaine_binaire=texte_vers_binaire(text).encode('utf-8')
    print(chaine_binaire)
    arduino=serial.Serial("COM8",baudrate=9600,timeout=.1)
    for i in range (len(chaine_binaire)):

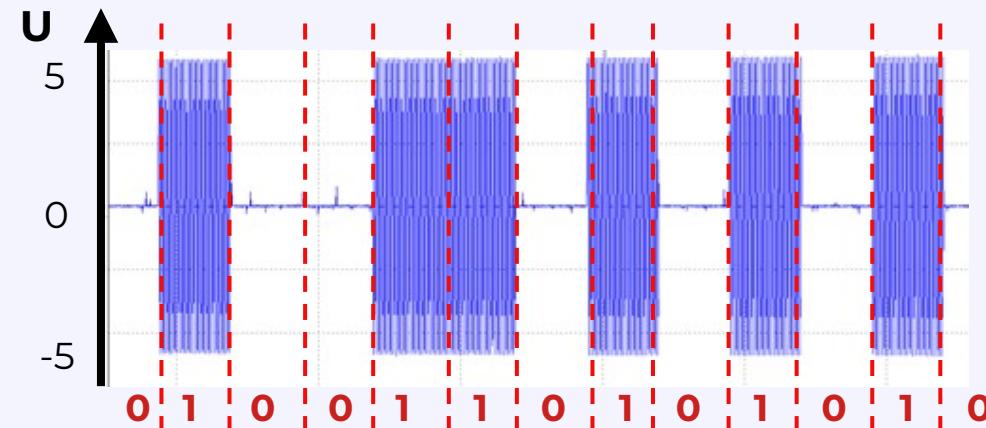
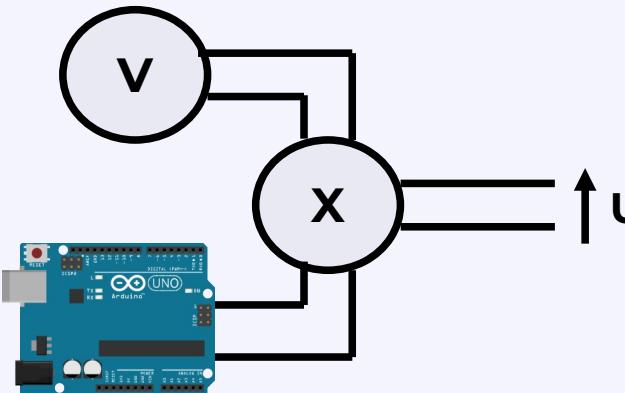
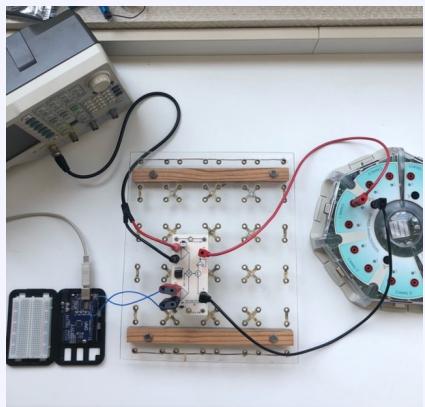
        arduino.write((chaine_binaire[i]))
```

Sortie Moniteur série x

0100110101010000



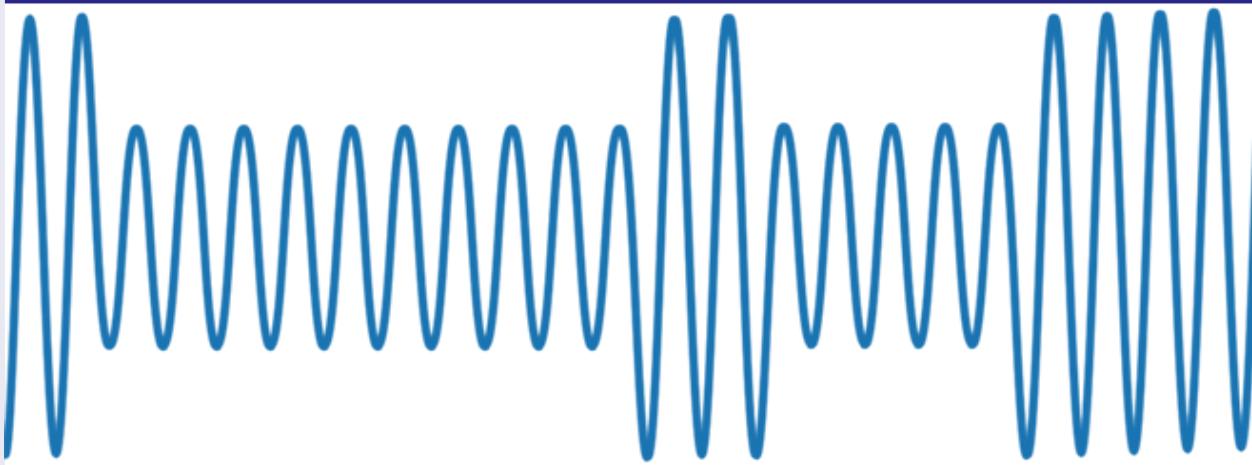
Avec multiplicateur :



III / Transmission de notre propre message

Réception du message :

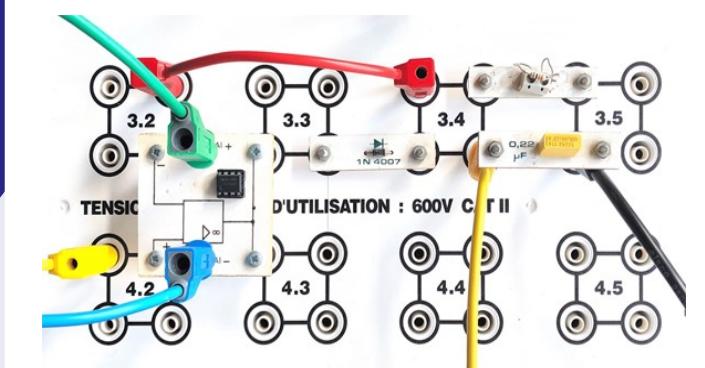
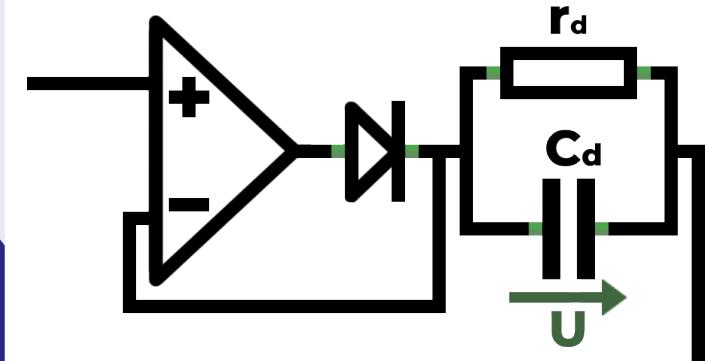
Signal observé dans le lecteur :



Signal porteur de l'information :



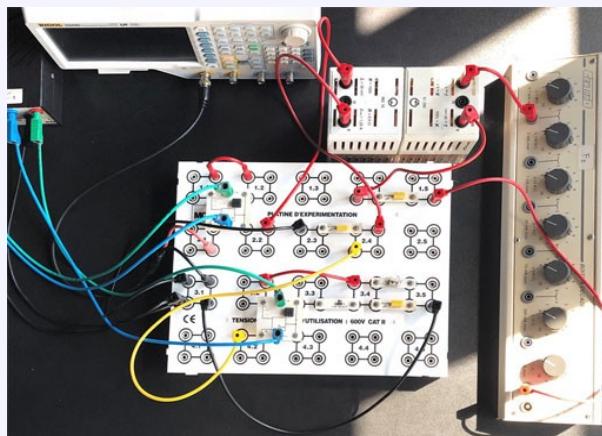
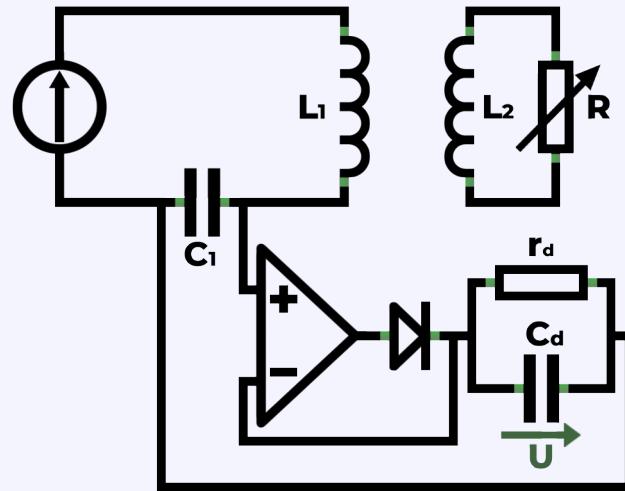
Démodulation :



II / Modélisation de la liaison carte-lecteur

Démodulation d'amplitude :

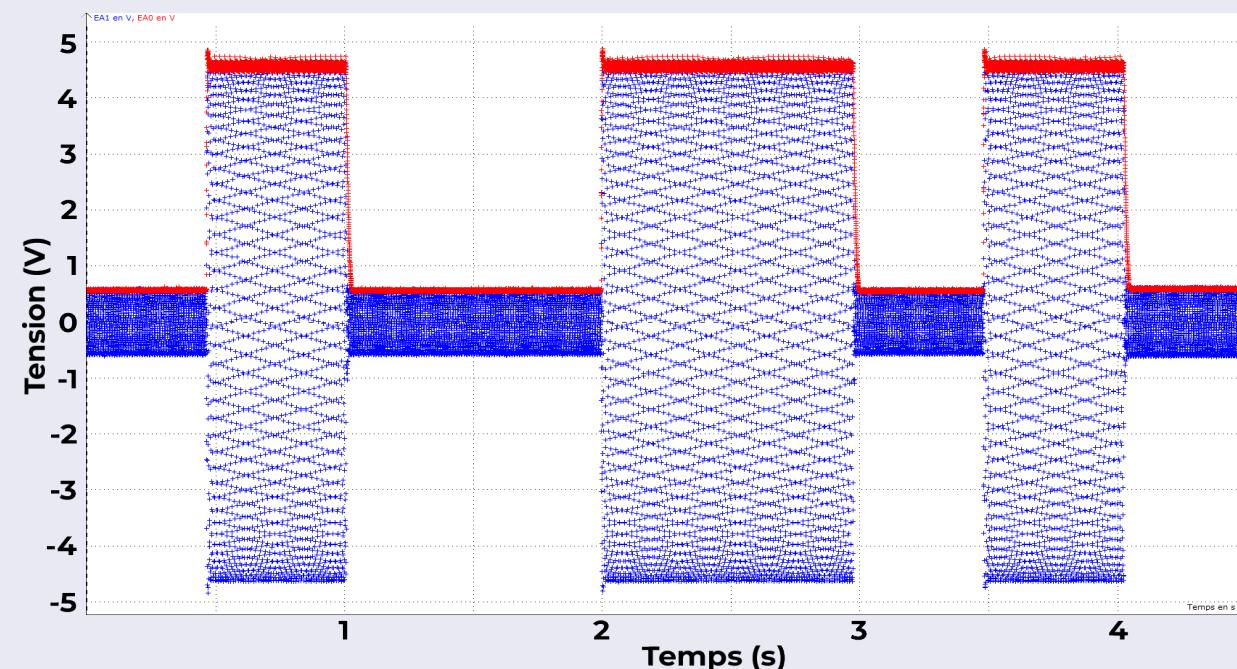
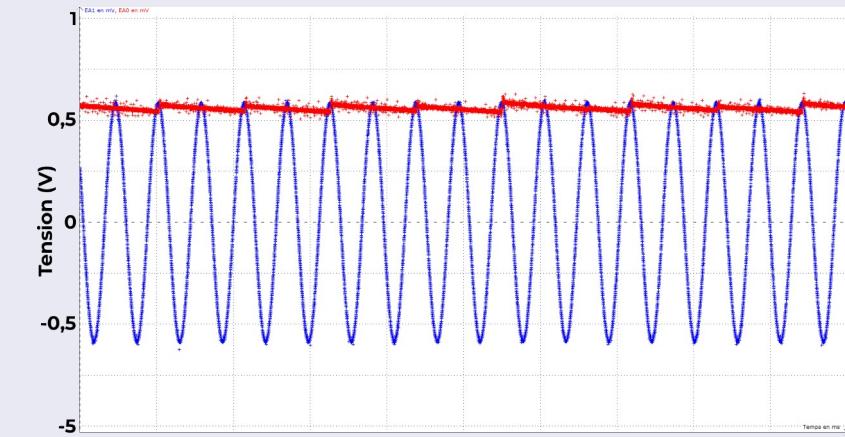
Dispositif :



Caractéristiques :

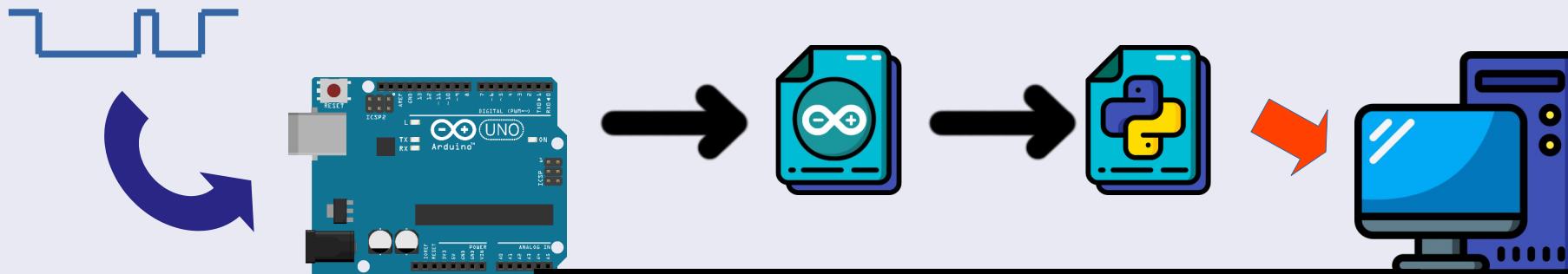
$$r_d = 10\text{k}\Omega$$

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



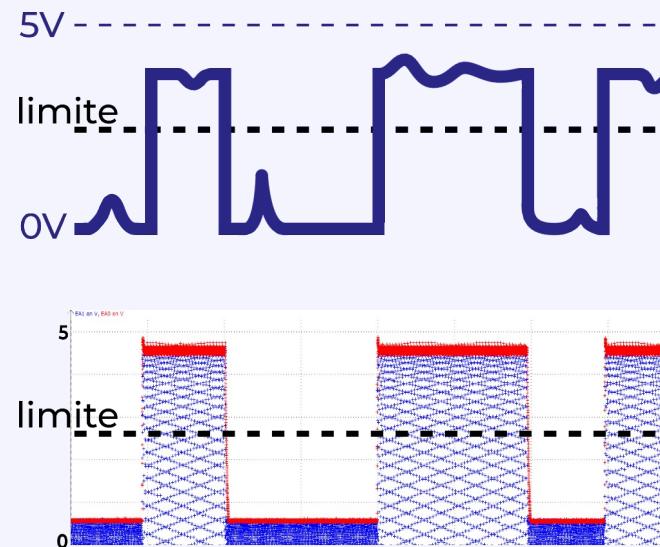
III / Transmission de notre propre message

Réception du message :



Résultat :

```
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 }
```



```
In [47]: runfile('D:/Spe MP CaJu/TIPE/code/code_re  
cleaned_data : [0, 0, 0, 0, 0, 1021, 0, 1021, 0,  
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  
binaire_data : [0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,  
message : MP
```

CONCLUSION

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

Annexes : partie envoi

code_arduino_send.ino

```
1 float messagerecubinaire;
2 void setup() {
3     Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
4     pinMode(13,OUTPUT);
5 }
6 void loop(){
7     if (Serial.available()>0){
8         messagerecubinaire=Serial.read(); //sinon Serial.parseInt()
9         //messagerecubinaire=messagerecubinaire - 48;
10
11
12         ;if (messagerecubinaire=='0') {
13             digitalWrite(13,LOW);
14             delay(50);
15         }
16
17         else if (messagerecubinaire=='1'){
18             digitalWrite(13,HIGH);
19             delay(50);}
20
21 }
```

Annexes : partie envoi

```
def texte_vers_binaire(text):
    # Initialise une chaîne binaire vide
    binary_string = ""

    # Itère sur chaque caractère du texte
    for car in text:
        # Convertit le caractère en sa valeur ASCII
        ascii_val = ord(car)
        # Convertit l'ASCII en binaire
        binary_val = bin(ascii_val)[2:]

        # ajoute des zéros si la valeur binaire est de moins de 8 bits
        while len(binary_val) < 8:
            binary_val = "0" + binary_val

        # ajoute la valeur binaire à la chaîne binaire
        binary_string += binary_val

    return binary_string
```

```
import serial
def texteversArduino(text):
    chaîne_binaire=texte_vers_binaire(text).encode('utf-8')
    print(chaîne_binaire)
    arduino=serial.Serial("COM8",baudrate=9600,timeout=.1)
    for i in range(1000):
        arduino.write(chaîne_binaire)
```

Annexes : partie réception

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 : partie réception

```
"""Initialisation des variables"""
tension_max = 5      # valeur max en Volt enregistrée par l'arduino
tension_low = 0       # valeur max en Volt lorsque la tension est basse
tension_mid = 2.5     # valeur en Volt du seuil

limit = int(tension_mid * 1023/tension_max)

nb_vals = 40

rawdata = []          # données brutes

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

Annexes : partie réception

```
def compare(L:list)->list:  
    """ arg : liste de tensions entre 0 et 1023  
    |   résultat : liste des valeurs binaires"""  
    result=[]  
    for elt in L :  
        if elt > limit :  
            result.append(1)  
        else :  
            result.append(0)  
    return newL  
  
def traduit(L:list)->str:  
    """ arg : L=[0,1,0, ...]  
    |   résultat : texte correspondant"""  
    result = ""  
    octets = [L[i:i+8] for i in range(0, len(L), 8)]  
  
    for octet in octets:  
        ascii_val = 0  
  
        for i in range (8):  
            ascii_val += octet[7-i]*(2**i) #calcul du chiffre codé par l'octet  
  
        char = chr(ascii_val) #transforme le code ascii en caractère  
        result += char  
  
    return result
```

Annexes : partie réception

```
def traduit(L:list)->str:  
    """ arg : L=[0,1,0, ...]  
        result : texte correspondant"""  
    result = ""  
    octets = [L[i:i+8] for i in range(0, len(L), 8)]  
  
    for octet in octets:  
        ascii_val = 0  
  
        for i in range (8):  
            ascii_val += octet[7-i]*(2**i) #calcul du chiffre codé par l'octet  
  
        char = chr(ascii_val) #transforme le code ascii en caractère  
        result += char  
  
    return result  
  
def write(L:list):  
    """inscrit les valeurs 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()
```