La RFID :

Comment les informations d'une carte RFID sont-elles réparties dans le message transmis au lecteur ?

I – Introduction à la RFID et détails de ma carte

- Utilisations courantes de la RFID
- Détermination du protocole de ma carte

II – Mise en place d'un lecteur automatique

- Un circuit électronique pour lire
- Un programme pour décoder

Dénominations usuelles :

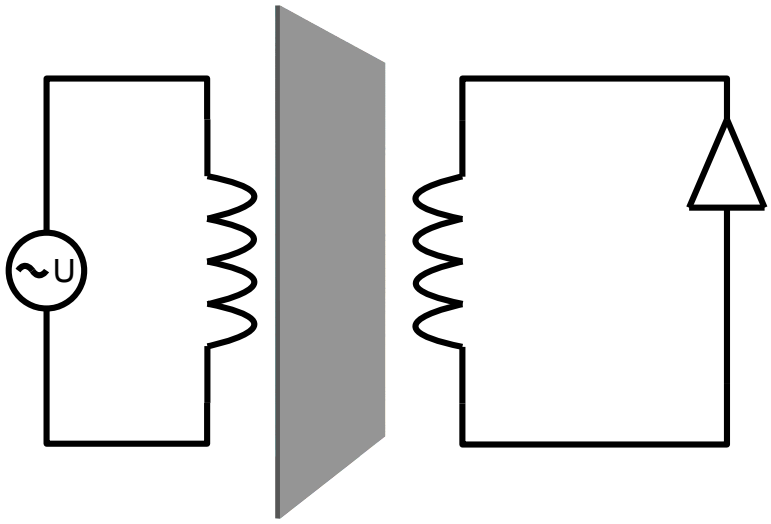
Nom commercial	Caractéristiques	Utilisation
NFC	13,56 MHz	Paiement sans contact
	≈ 1 cm	Cartes de transport
	≈ 10 ko/s	authentification
	125 kHz	Suivi de paquets
	≈ 1 cm	Badges de paiement
	100 o/s	Identification des animaux
	8,2 MHz	Anti-vol

Notes :

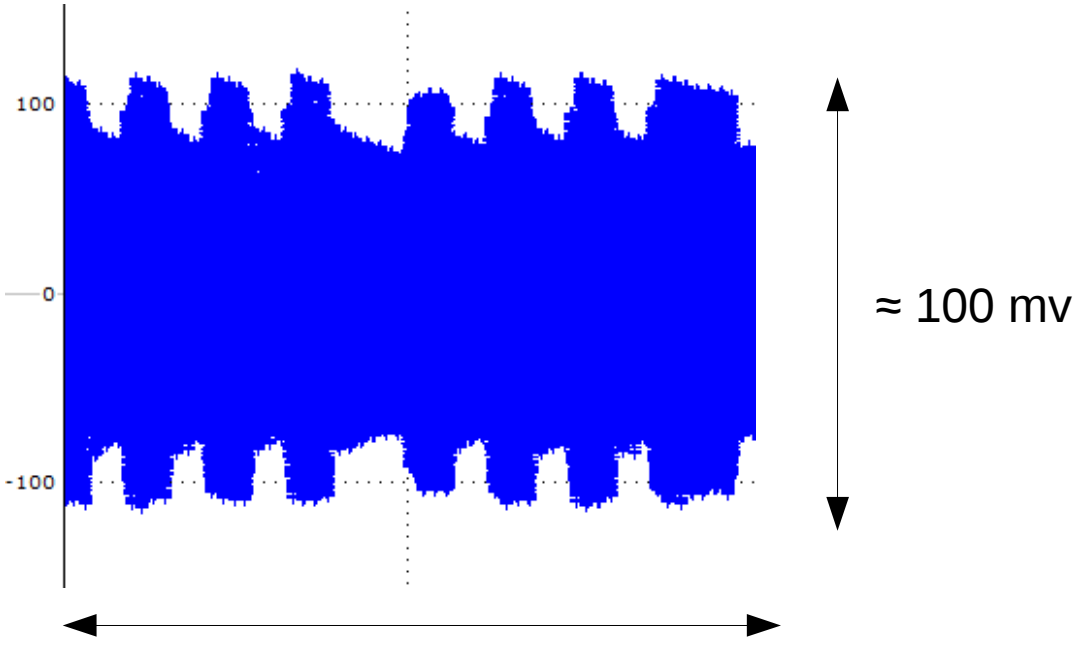
- Différentes fréquences : pénétration / vitesse / disponibilité
- Communication uni/bi-directionelle
- Programmation à distance

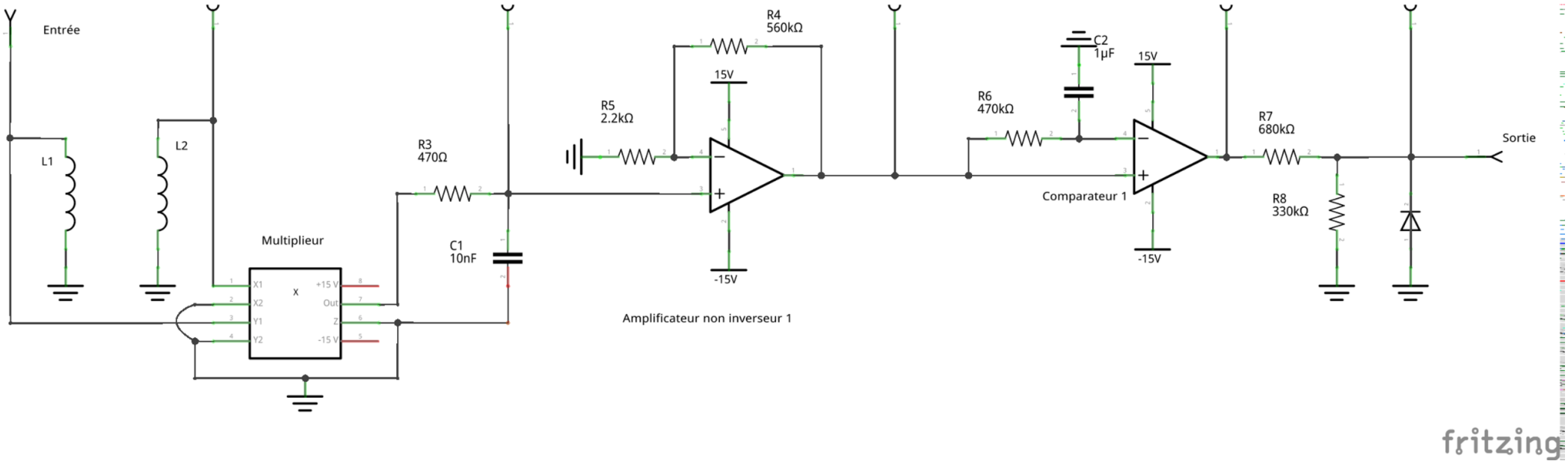
Analyse de ma carte :

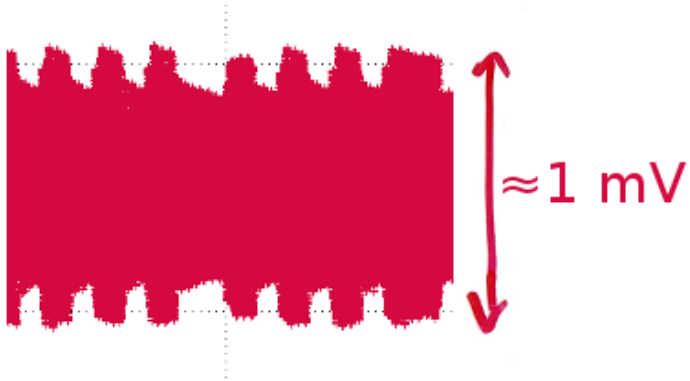
- Mesure approximative
- Approche plus précise

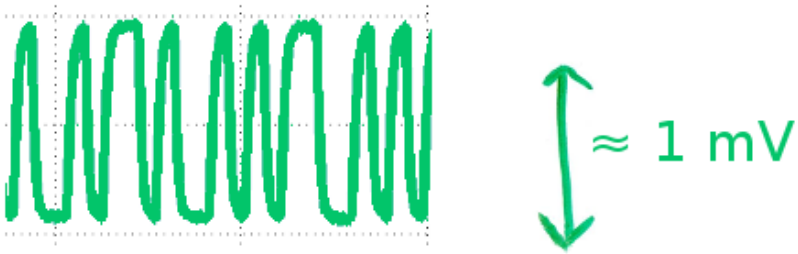


à 134kHZ :



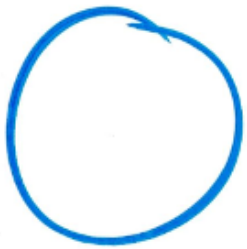

Entrée
L1
L2
Multiplieur
X1
X2
Y1
Y2
X
+15 V
Out
Z
-15 V
R3
470Ω
C1
10nF
R5
2.2kΩ
R4
560kΩ
15V
-15V
Amplificateur non inverseur 1
R6
470kΩ
C2
1µF
15V
-15V
Comparateur 1
R7
680kΩ
R8
330kΩ
Sortie
fritzing



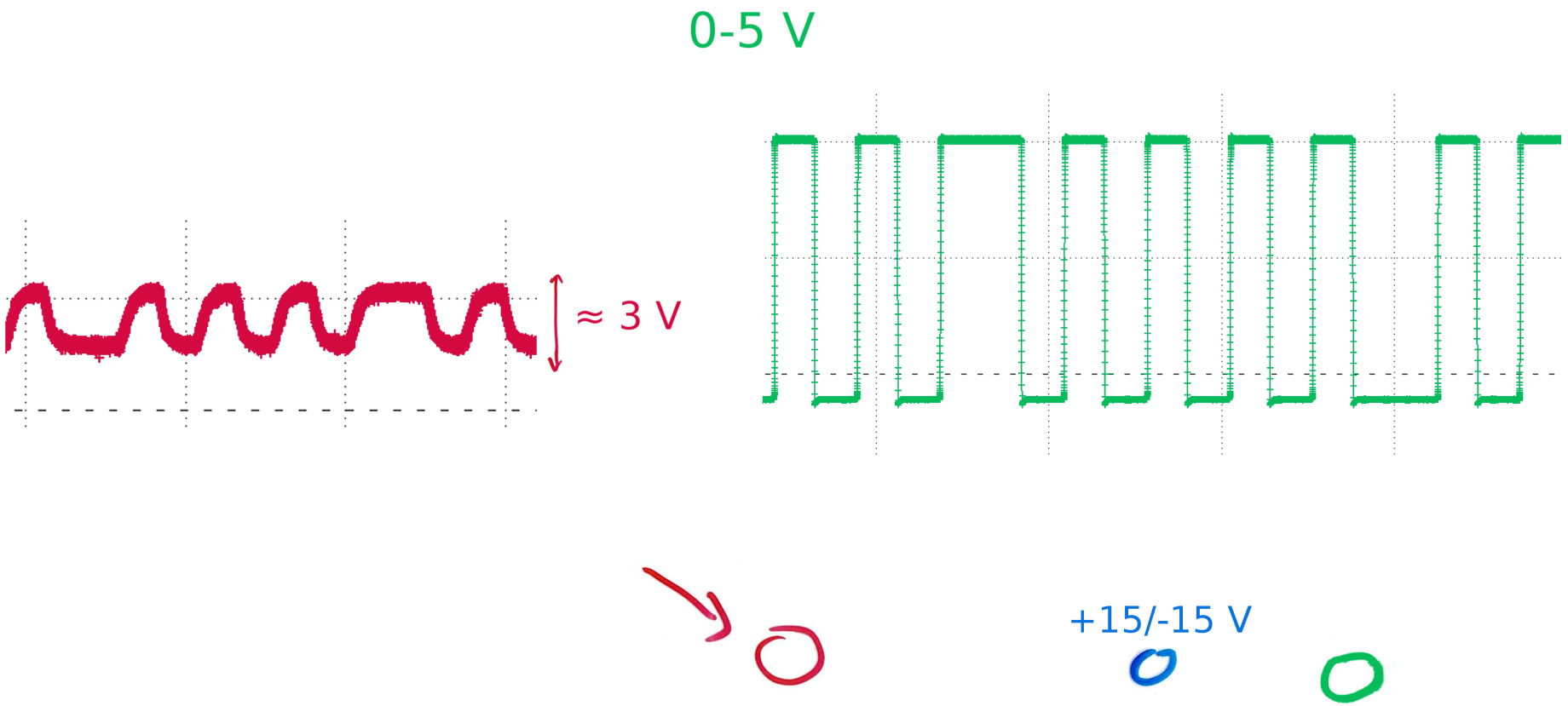


+ composante continue

$(A+s(t))\cos(ft)^2=(1+s(t))/2\ (1+\cos(2ft))$



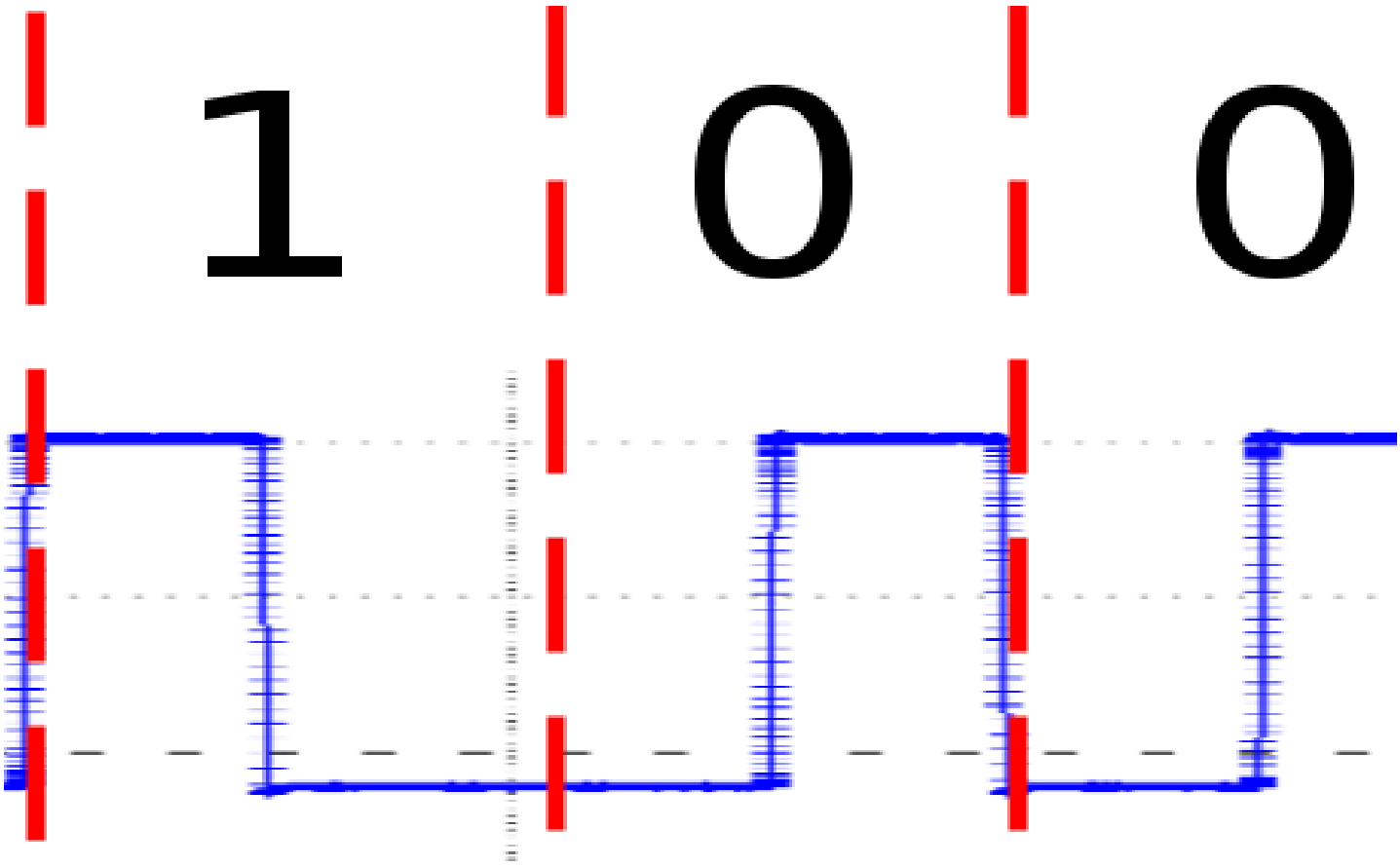
Passe bas
supprime le terme cos(2ft)



Amplification de 1+R4/R5 ≈ 300

Détails du protocole de ma carte :

- Codage en Manchester

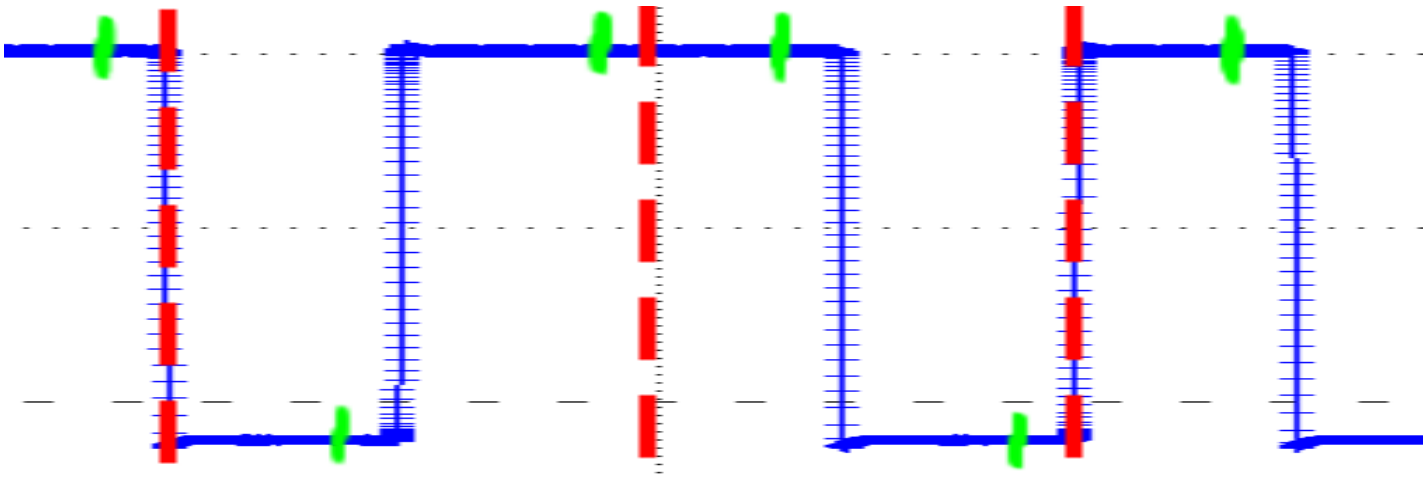


Remarque : une impulsion longue indique un changement de bit

Analyse de ma carte :

- Première idée

  On effectue UNE seule mesure par période



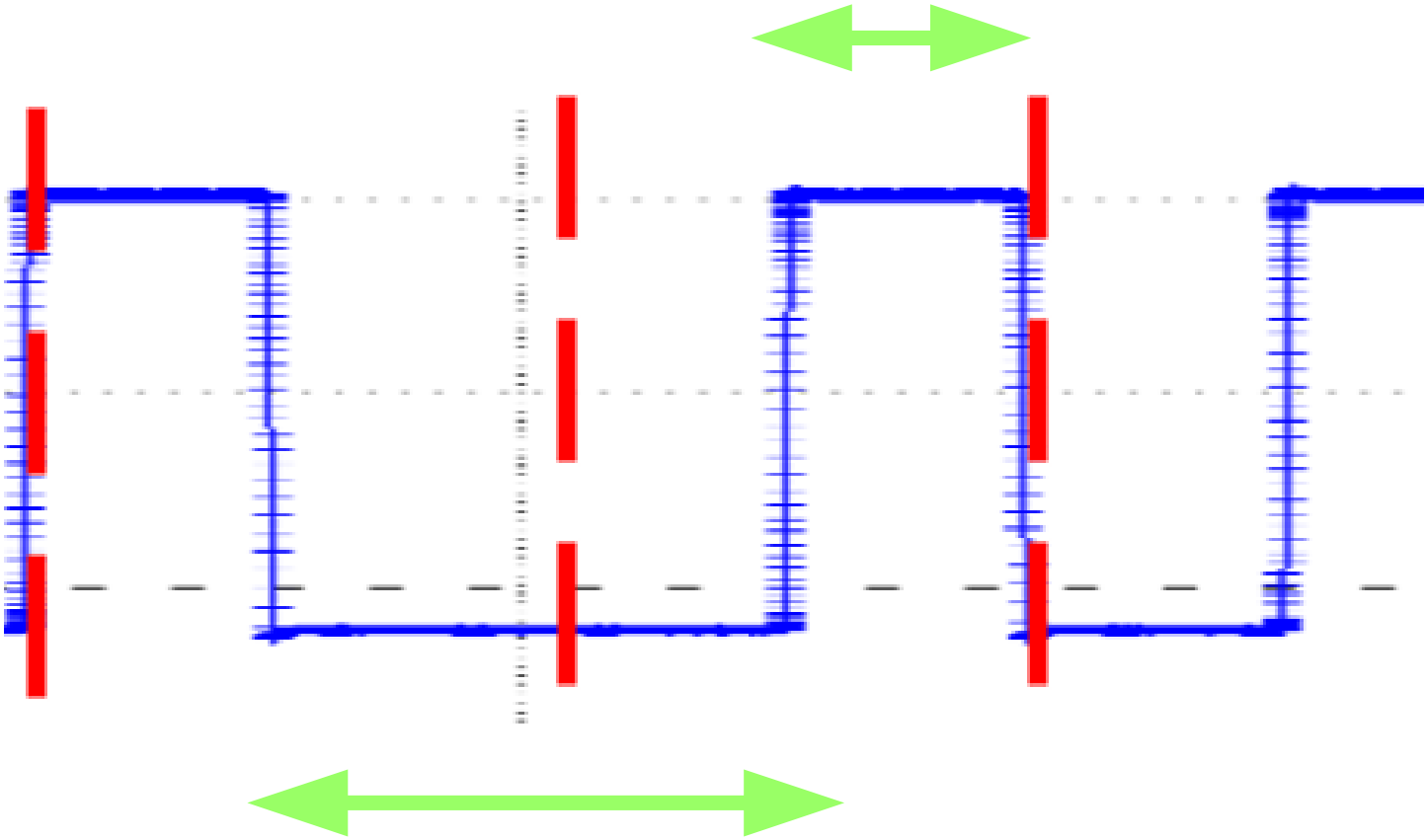
Échec car peu précis

Analyse de ma carte :

- Seconde idée

  On mesure les caractéristiques de chaque impulsion

209 µs - haut



449 µs - bas

Méthode précise et efficace

Décodage par ordinateur :

- Détail du protocole

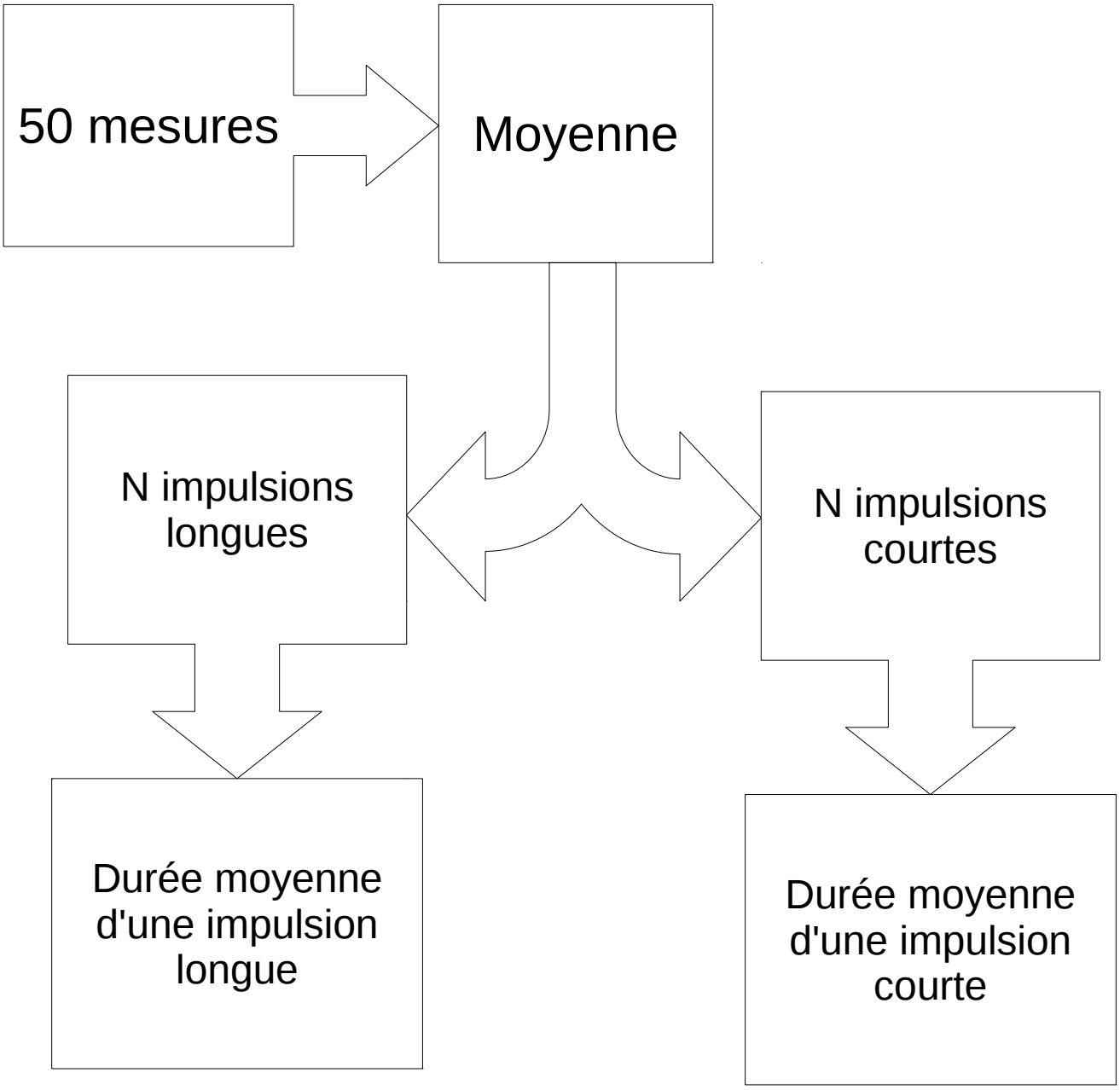
  EM4100 (EM Microelectronic)

1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
				0	**0**	**0**	**0**	**0**
				1	**0**	**1**	**0**	**0**
				0	**0**	**0**	**0**	**0**
				0	**0**	**1**	**1**	**0**
				0	**0**	**0**	**0**	**0**
				0	**0**	**0**	**0**	**0**
				0	**0**	**0**	**0**	**0**
				0	**0**	**0**	**1**	**1**
				1	**1**	**0**	**0**	**0**
				0	**0**	**1**	**1**	**0**
				0	**1**	**1**	**1**	**0**

Décodage par ordinateur :

- Détail du programme

    1) Distinguer impulsions longues et courtes

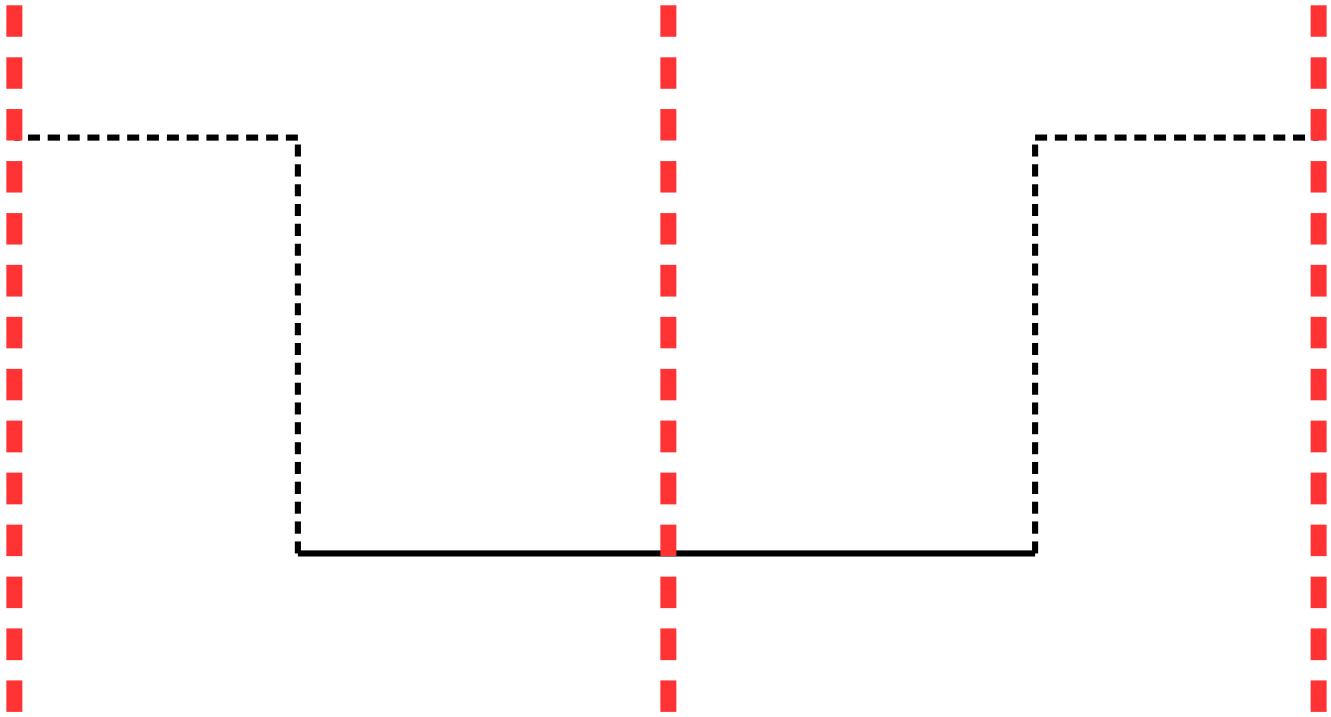


Ensuite on compare à la moyenne des deux durées

Décodage par ordinateur :

2) Reconstituer les bits par récurrence

a) Initialisation

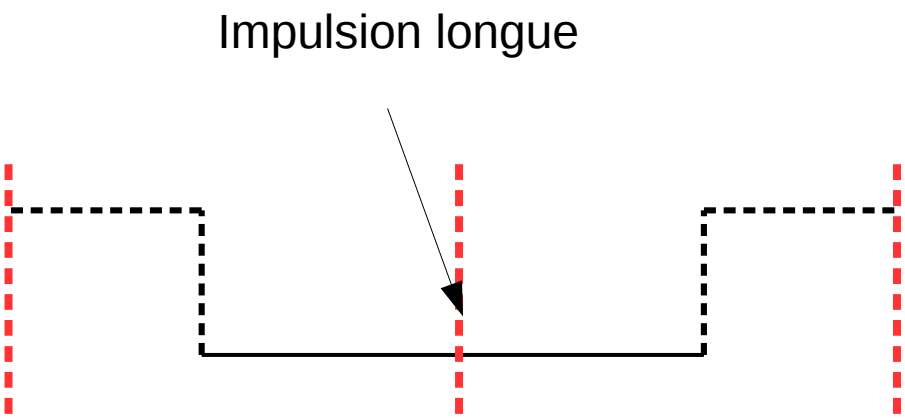


0 1

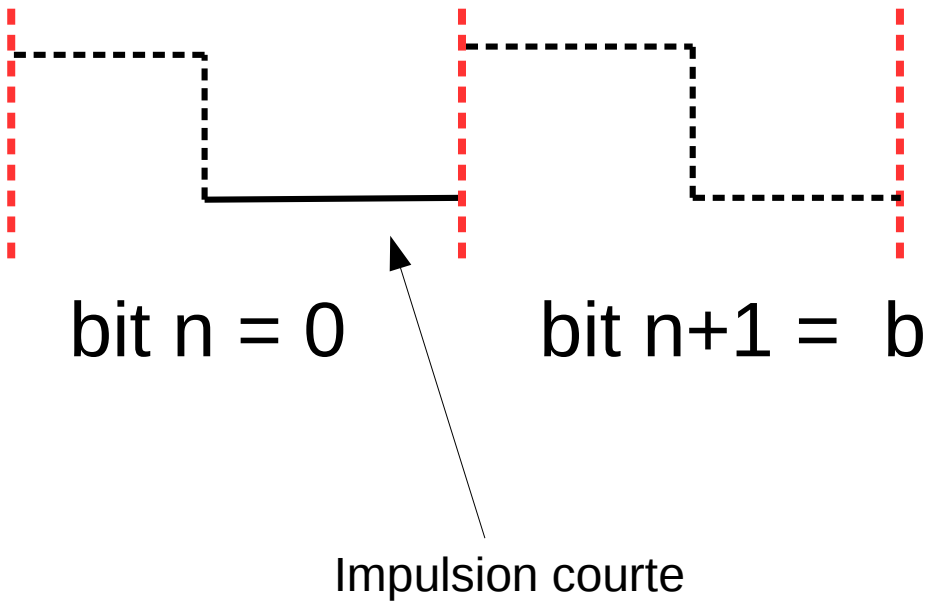
Décodage par ordinateur :

b) Hérédité

Cas ou le bit n est 0



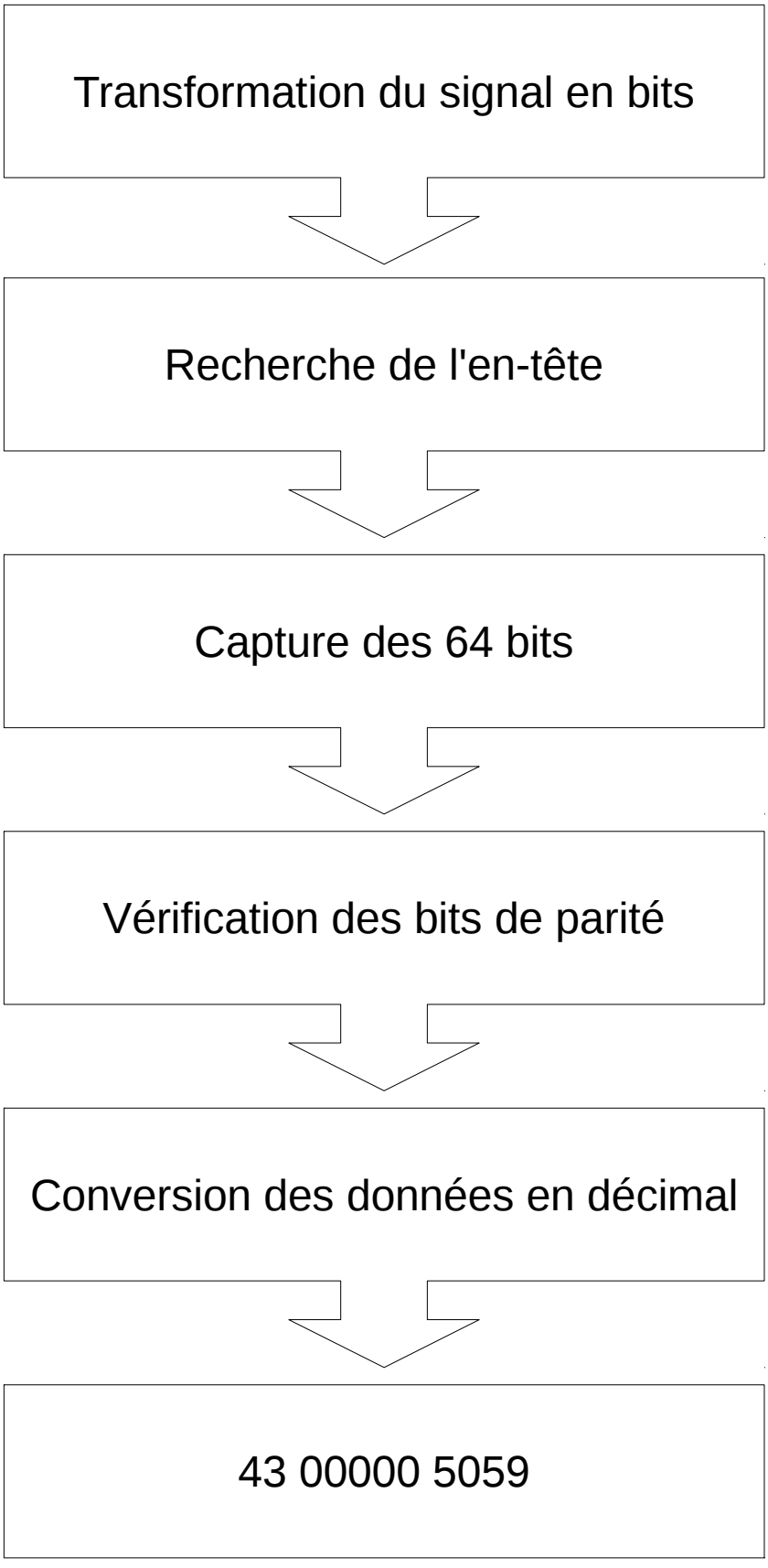
bit n = 0 bit n+1 = 1



bit n = 0 bit n+1 = bit n

Décodage par ordinateur :

3) Reconstituer le message



Bilan et conclusions :

- Peu d'informations dans la carte
- Aucune protection des données

Reconstitution des bits :

```
def read(ser):
    echantillon = []
    ser.flushInput()
    ser.readline()
    for i in range(50):
        duree = int(ser.readline().decode().replace("\r\n","")[2:])
        echantillon.append(duree)
    moy = mean(echantillon)
    longs = [a for a in echantillon if a > moy]
    courts = [a for a in echantillon if a < moy]
    court = mean(courts)
    long  = mean(longs)
    moy = mean([court,long])
 #---------------------------------------------------------------------------
    bits=[]
    message = ser.readline().decode().replace("\r\n","")
    while message[0] != 'l' or int(message[2:]) < moy:
        message = ser.readline().decode().replace("\r\n","")
    # On a alors un 1
    Bits+=[1,0]
# ---------------------------------------------------------------------------
    # On va ensuite décoder les 200 bits suivants
    for i in range(200):
        print(i)
        duree = int(ser.readline().decode().replace("\r\n","")[2:])
        last = bits[-1]
        if duree > moy:
            # On a une pulsation longue donc on passe au bit
opposé
            bits.append(int(not last))
        else :
            # On a une pulsation courte donc on conserve le
même bit
            bits.append(last)
            ser.readline()
    return bits
```

Evaluation des durées des impulsions

Initialisation

Récurrence

La fonction retourne une liste contenant environ 200 bits

Décodage :

```
def find_header(l):
    check = [1]*9
    init = None
    for i in range(0,len(l)-64):
        test = [l[k] for k in range(i,i+9)]
        if test == check :
            init = i+1
            break
    return l[init+8:init+63]
```

```
def check(l):
    arr = np.array(np.array_split(l,11))
    for ligne in arr[:-1 ]:
        check = sum(ligne[:4])%2
        assert check == ligne[-1]:
    message = arr[:-1,:-1]
    return message
```

```
def decodage(l):
    bits = ''
    for ligne in l:
        bits += ''.join(ligne.tolist())
    return int(bits,2)
```