



il faut un détecteur d'enveloppe :

$$\textcircled{1} \text{ diode bloquée } \frac{ds}{dt}(t) + \frac{1}{G}s(t) = 0 \text{ et } s(t) > e(t)$$

$$\textcircled{2} \text{ diode passante } s(t) = e(t) \text{ et vérifier que } \frac{ds(t)}{dt} + \frac{1}{G}s(t) > 0$$

$$S[i] = s(t_i)$$

$$Q3: \quad \frac{ds}{dt}(t_i) = \frac{s(t_{i+1}) - s(t_i)}{t_{i+1} - t_i}$$

relation de récurrence quand la diode est bloquée à l'instant t_i :

$$\frac{ds(t)}{dt} + \frac{1}{G}s(t) = 0$$

$$\frac{s(t_{i+1}) - s(t_i)}{\Delta t} + \frac{1}{G}s(t_i) = 0$$

$$\boxed{s(t_{i+1}) = s(t_i) \left(1 - \frac{\Delta t}{G}\right)}$$

Q4 Diode passante en t_i , donc la relation est $s(t) = e(t)$

la valeur de $s(t_i) = e(t_i)$, en suivant cette démarche $s(t_{i+1}) = e(t_{i+1})$ et on ne vérifie pas que la relation est devenue nulle., il nous faudrait la valeur suivant t_{i+2}

$$\frac{ds(t_{i+1})}{dt} + \frac{1}{G}s(t_{i+1}) = 0$$

$$\frac{s(t_{i+1}) - s(t_i)}{\Delta t} + \frac{1}{G}s(t_{i+1}) = 0$$

$$s(t_{i+1}) \left(\frac{1}{\Delta t} + \frac{1}{G} \right) = \frac{s(t_i)}{\Delta t}$$

$$\frac{ds(t_{i+1})}{dt} = \frac{s(t_{i+1}) - s(t_i)}{\Delta t}$$

erreur

(ds corrigé)

$$\boxed{s(t_{i+1}) = \frac{\frac{\Delta t}{G} + 1}{\Delta t + G} s(t_i)}$$

Q5: def solve(T, E, tau):

$$S = [\phi] * \text{len}(T) \quad \text{delta_t} = [T[1] - T[0]]$$

diode = "passante"

~~if diode == "passante" :~~

for i in range(len(T)):

~~if diode == "passante" :~~

$$S[i] = E[i], S[i+1] = \frac{\tau}{\tau + \Delta t} S[i]$$

~~if S[i] <= E[i] : S[i+1] = E[i]~~

~~else : S[i+1] = S[i] + (E[i] - S[i]) * delta_t / tau~~

erreur de
corrige

else :

$$S[i] = S[i-1] * (1 - \frac{\Delta t}{\tau})$$

$$S[i+1] = S[i] * (1 - \frac{\Delta t}{\tau})$$

~~if S[i] <= E[i] :~~

 diode == "passante"

return (S)

Q6:

$$\Delta t_1 = 100 \text{ ns}$$

$$\Delta t_1 = 0,1 \cdot 10^{-6} \text{ donc } 30 \text{ mesures}$$

erreur de
corrige

$$\Delta t_2 = 10 \text{ ns}$$

$$\Delta t_2 = 0,01 \cdot 10^{-6} \text{ donc } 300 \text{ mesures}$$

$$\Delta t_3 = 1 \text{ ns}$$

$$1 \text{ ns} = 1 \cdot 10^{-9} \text{ s } 3000 \text{ mesures}$$

$$T_{\max} = 0,000003 = 3 \cdot 10^{-6} \text{ s}$$

Q7: cas $\Delta t_1 = 100 \text{ ns}$, euler explicite diverge ... résultats aberrants.

Cas $\Delta t_2 = 10 \text{ ns}$, une zone incertaine

cas $\Delta t_3 = 1 \text{ ns}$, répond exactement aux conditions.

Q8:

Réultat 1: inexploitable

Réultat 2: écart trop grand pour avoir 1,3V et 1,5V

Réultat 3: On distingue clairement le maxi qui est toujours le même sur un intervalle

$\Delta 0,2\text{V}$ qui peut être vérifié.

Variation de $s(t)$ dans un intervalle de $0,1\text{V}$.

$$T = 1\mu\text{s} \quad \frac{\Delta t}{T} \text{ petit} \quad \frac{1\text{ns}}{1\mu\text{s}} = \frac{10^{-9}}{10^{-6}} = 10^{-3} \text{ facteur } 1000$$

Signal RFID 13,56

2.1. bit de parité.

Q9:

$$(5)_{10} = (0101)_2 \Rightarrow \text{bit de parité} = 0$$

$$(18)_{10} = (10000)_2 \Rightarrow \text{bit de parité} = 1$$

$$(37)_{10} = (100101)_2 \Rightarrow \text{bit de parité} = 1$$

Erreurs de
corrigé

Q10: def parity(bits):



"bits est une chaîne de liste"

S = 0

for b in bits:

$$S = S + b$$

if S % 2 == 0:

return 0

return 1

else:

return 1

Q11: - permutation d'un 0 et un 1 (pas de détection)

- pas de correction possible car emplacement erroné non déterminé.

Code de Hamming :

code (7,4) \rightarrow 3 bits de parité pour 4 bits de données

($p_1, p_2, d_1, p_3, d_2, d_3, d_4$)

Q12: def encode_hamming(donnee):

$p_1 = \text{parité}([d_1, d_2, d_3])$

$p_2 = \text{parité}([d_1, d_2, d_4])$

$p_3 = \text{parité}([d_1, d_3, d_4])$

return ($[p_1, p_2, d_1, p_3, d_2, d_3, d_4]$)

Contôle après réception :

$$[c_1, c_2, c_3] = [0, 0, 0]$$

Q13: def decode_hamming(message): si $c_1=0$ pb sur d_1

$c_1 = \text{parité}([m_4, m_5, m_6, m_7]) = \text{parité}([p_3, d_2, d_3, d_4])$

$c_2 = \text{parité}([m_1, m_2, m_3, m_7]) = \text{parité}([p_2, d_1, d_3, d_4])$

$c_3 = \text{parité}([m_1, m_3, m_5, m_7]) = \text{parité}([p_1, d_1, d_2, d_4])$

if $[c_1, c_2, c_3] == [0, 0, 0]$:

autre solution

return $[d_1, d_2, d_3, d_4]$

position_error = $c_1 * 4 + c_2 * 2 + c_3$

else:

if $[c_1, c_2, c_3] == [1, 0, 0]$:

print "erreur sur le bit d_1 "

$d_1 = \text{int}((\text{bin}(d_1) + \text{bin}'(1))(\phi))$

if $[c_1, c_2, c_3] == [0, 1, 0]$:

print "erreur sur le bit d_2 "

$d_2 = \text{int}((\text{bin}(d_2) + \text{bin}'(1))(\phi))$

if $[c_1, c_2, c_3] = [1, 0, 1]$:

print "erreur sur le bit d2"

if $[c_1, c_2, c_3] = [1, 1, 1]$:

print "erreur sur le bit d4"

return $[d_1, d_2, d_3, d_4]$

Q14:

donnée (1011)

encode_hamming(1011) = (0, 1, 1, 0, 0, 1, 1)

transmission ~~(0, 1, 0, 1, 1, 1, 1)~~

$c_1 = 0$ $c_2 = 1$ $c_3 = 1$ $(1, 0, 1, 0, 0, 1, 1)$

la correction (faite dans le programme précédent)

~~(1, 0, 0, 0, 1, 0, 1)~~ ↗

1 bit ~~d₃~~ est faux. $(1, 0, 0, 0, 0, 1, 1)$
alors qu'il n'y avait pas d'erreur
message $(0, 0, 1, 1)$

Q15:

bit de parité sur les bits de contrôle

non!

non, puisque inversion de p₁ et p₂

utilisation des données de la puce pour autoriser ou non le passage

Q16 id_titre : entier (490 87 654)

zones : liste [1, 3]

date_fin : liste [année, mois, jour]
2015 08 31

Q17.

```

fichier = open('tableau.csv', 'w')
fichier.write('indice'; + '0' + ; + '1' + ... \n)
fichier.write('contenu' + ; +
for i in range(2, 5):
    donnees = ligne[i].rstrip('\n').split(',')
    temps = donnees[0].split('-')
    fichier.write('contenu' + ; + temps[0]... )

```

Q18 : def estAvant (date1, date2) :

 autre if date1[0] < date2[0] : Res = False

 if date1[2]<date2[2]: return (True) Res = True

 if date1[1]<= date2[1]: if date1[0] == date2[0] :

 if date1[1]<= date2[1]: return True if date1[1]< date2[1]:

 else: return False return (True) Res = True

 else: if date1[1]== date2[1] :

 if date1[2]< date2[2] : return (True) Res = True

 return (Res) else:

 return (False)

 else:

 return (False)

Q19 calcul de secondes .

Q20. def test_passage (donnees_carte):

 liste_noire

 -(date1, date2)