La modélisation mathématique de la propagation des virus :

TIPE : 2021-2022

THEME : Santé et prévention

BENDYA Adnane
Numéro d'inscription : 14562
Encadré par : Mr. RWAWI Said

Introduction :

- La propagation de plusieurs maladies épidémiques à travers l'histoire .
- Accélération de la propagation des pandémies due à l'évolution technologique (Transport).
- Les conséquences sanitaires et économiques.
- La compréhension de l'évolution épidémique est un élément crucial.
- La modélisation mathématique des pandémies.

Plan :

- Modèle SIR
- Modèle SEIR
- Application au cas de mon centre CPGE
- Programme informatique sur la gestion du nombre d'individus dans une salle

Modèle SIR :

Suceptible → Infected → Recovered

- **N=S(t)+I(t)+R(t)** : représente alors la population constante totale au cours du temps, on considère que le nombre nombre de naissance est constant (dans notre cas , on considère qu'il n'y a pas de nouveaux inscrits dans le centre CPGE)

Représentation du modèle SIR :

- β : représente le *taux de transmission.*
- γ : représente le taux de guérison

❖ *Les équations du modèle SIR :*

❖ $\frac{dS}{dt} = \Lambda - \beta(1 - \alpha)(1 - \delta)SI - \alpha S$

❖ $\frac{dI}{dt} = \beta(1 - \alpha)(1 - \delta)SI - \gamma I - \delta I$

❖ $\frac{dR}{dt} = \gamma I + \alpha S + \delta I$

- $0 \leq \alpha < 1$: le taux de confinement des susceptibles.
- $0 \leq \delta < 1$: le taux de l'isolation des infectieuses

❖ *Simplification des équations du modèle SIR :*

A l'aide des hypothèses du modèle SIR , le système d'équations devient :

- ✓ $\frac{dS}{dt} = -\beta SI$ **(1)**
- ✓ $\frac{dI}{dt} = \beta SI - \gamma I$
- ✓ $\frac{dR}{dt} = \gamma I$ **(3)**

❖ Détermination de l'expression de β et γ :

En intégrant la première et la troisième équation entre 0 et +∞ en vérifiant que S(∞) et R(∞) existent .

- **log S(+∞) – log S(0)= -$\beta \int I(t)dt$**
- **R(+∞) – R(0)= $\gamma \int I(t)dt$**

Les conditions initiales **: On a la population saine égale à la population totale S(0)=1 et R(0)=0 (au début de l'épidémie), alors nous obtenons :**

$$\beta = -\frac{\log(S(+\infty))}{\int_0^{+\infty} I(t)dt} \quad et \quad \gamma = \frac{R(+\infty)}{\int_0^{+\infty} I(t)dt}$$

Le taux de reproduction R0 :

- ***Définition :*** **Le taux de reproduction R0 est le nombre moyen de cas secondaires produits par un individu infectieux au cours de sa période d'infection.**
- ***Théorème du seuil :*** **Si R0>1, alors I(t) croît, atteint son maximum puis décroît vers 0 quand t tend vers +∞ : c'est une épidémie.**

Visualition :

- **Un cas qui illustre le théorème du seuil sur le modèle SIR :**

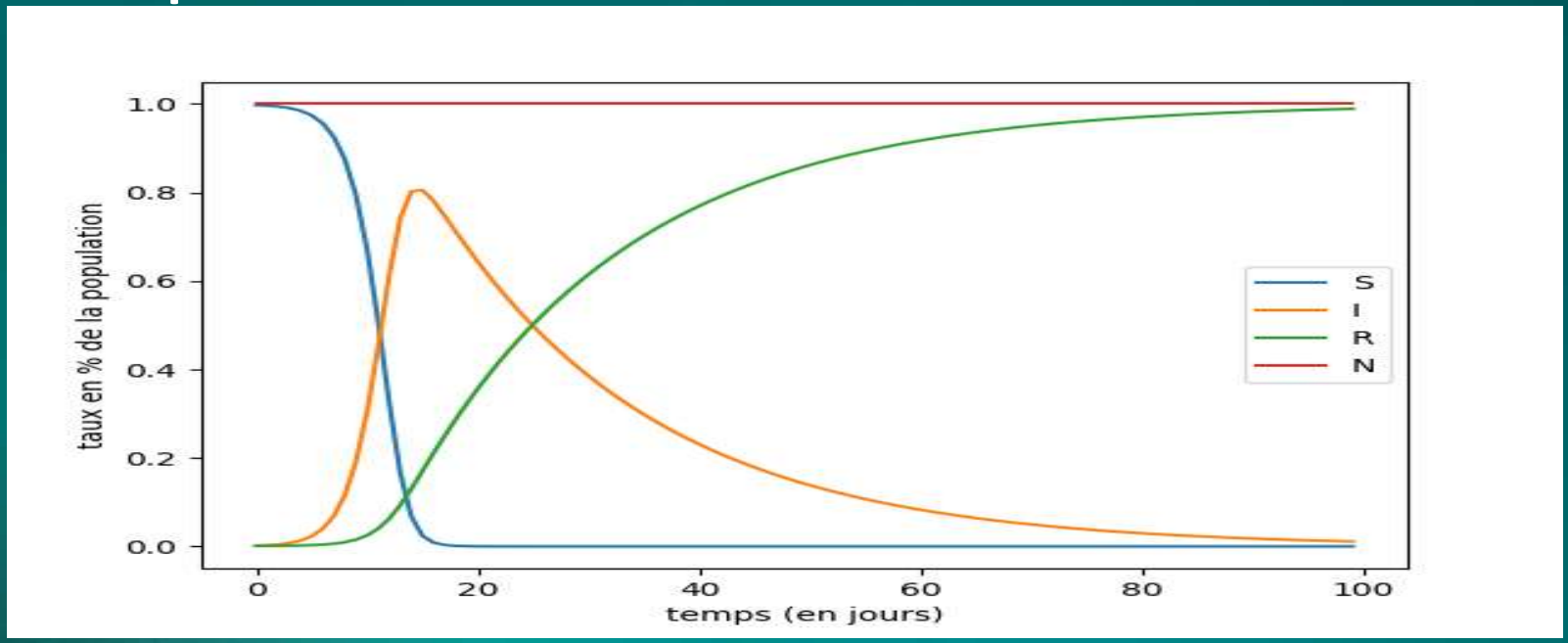


Figure 1 : Le taux de reproduction est de 16. De plus, on a pris 0,8 comme taux de transmission et 0,05 comme taux de guérison.

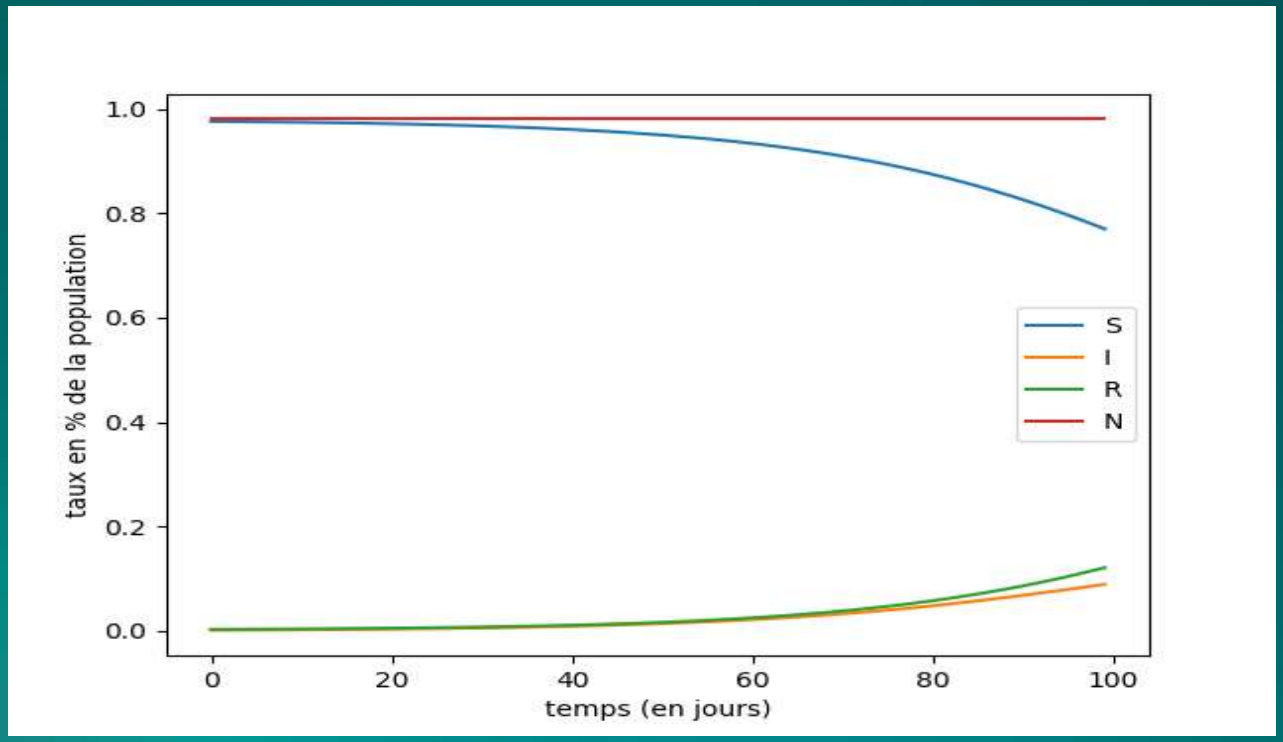


Figure 2: Le taux de reproduction R0 est de 2. De plus, on a pris 0,1 comme taux de transmission et 0,05 comme taux de guérison.

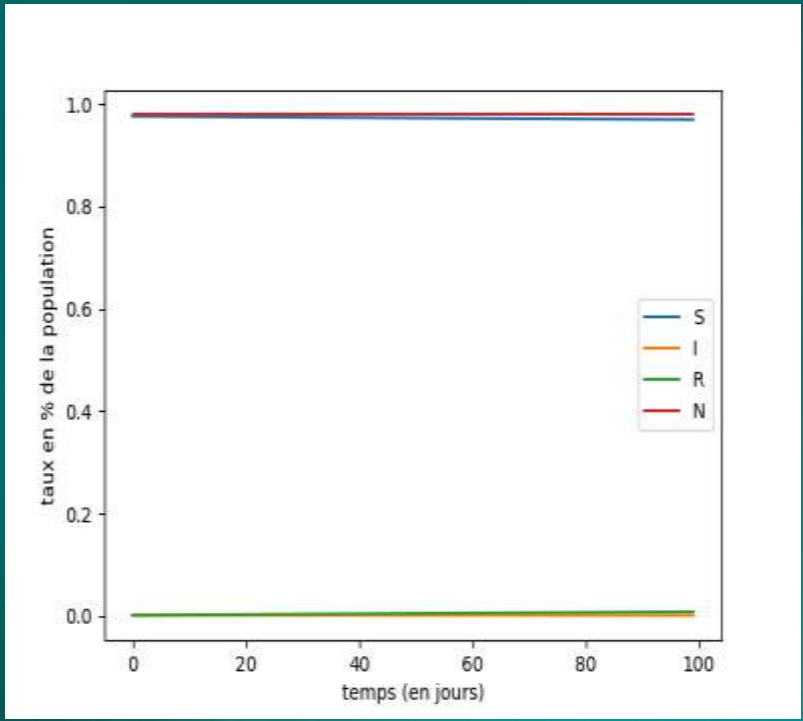


Figure 3 : Le taux de transmission est de 0,05 .Le taux de guérison est fixé à 0,05 , R0=1.

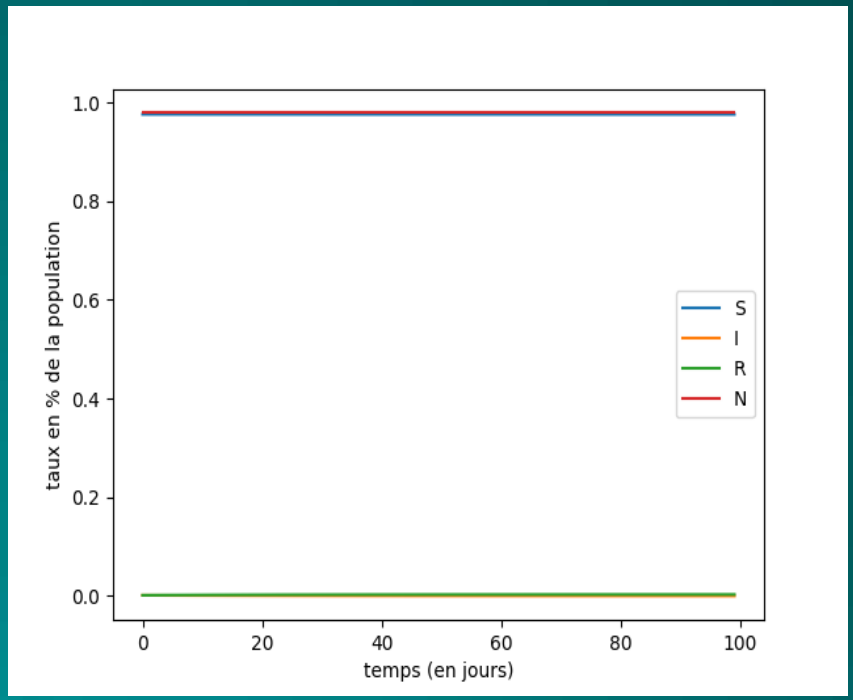
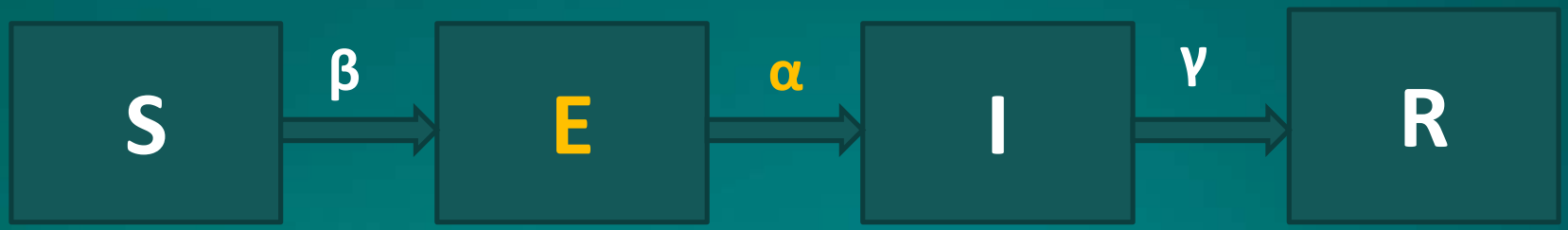


Figure 4 : Le taux de guérison est de 0,05 Le taux de transmission est fixé à 0,01.

Modèle SEIR :

- Ce modèle est une extension du modèle SIR qui instaure la possibilité que les personnes contaminées ne soient pas directement contagieuses, en ajoutant une classe de personnes dites exposées et notée E.



✓ On suppose que la population totale est constante au cours du temps :
$S(t) + I(t) + R(t)+E(t) = N$

Les équations :

- ✓ $\frac{dS}{dt} = -\beta S(t)I(t)$
- ✓ $\frac{dE}{dt} = \beta S(t)I(t) - \alpha E(t)$
- ✓ $\frac{dI}{dt} = \alpha E(t) - \gamma I(t)$
- ✓ $\frac{dR}{dt} = \gamma I(t)$

Visualisation :

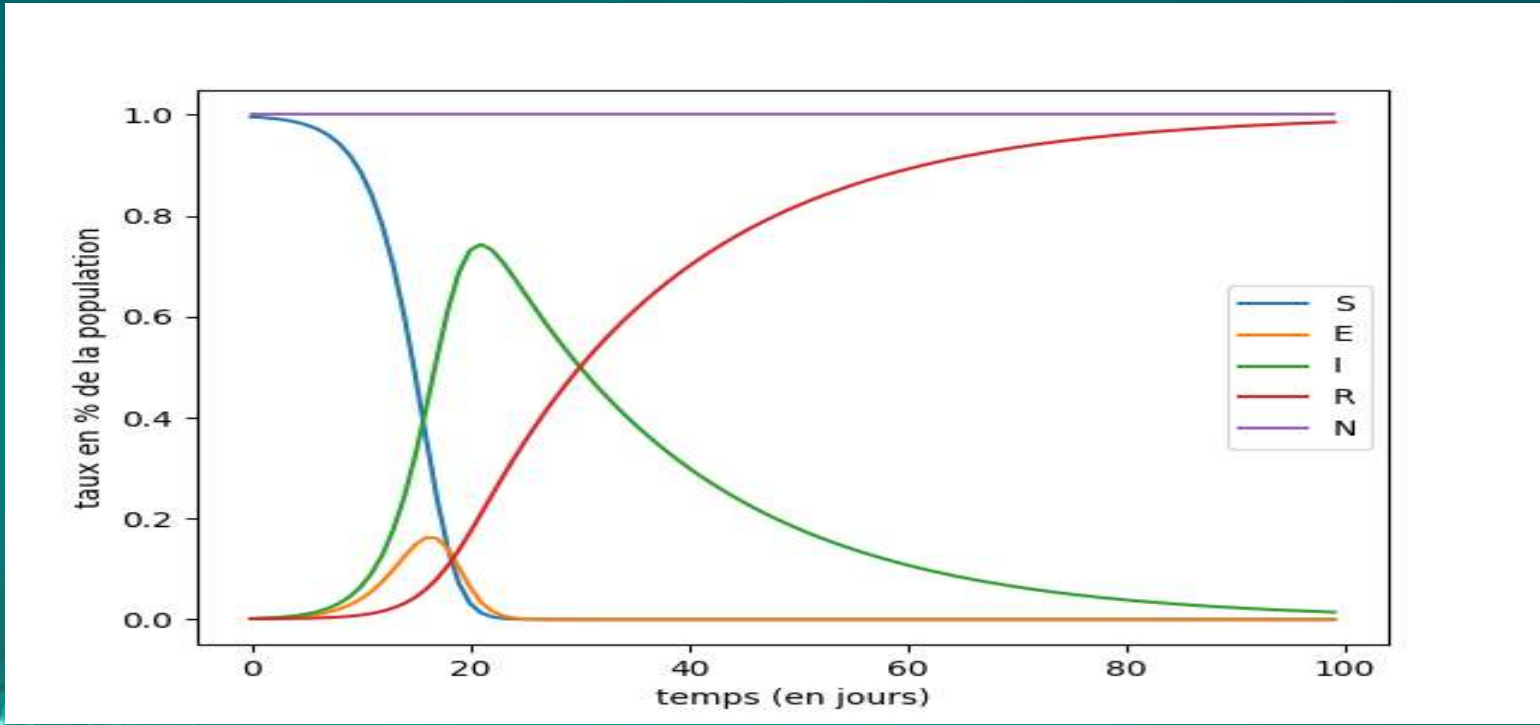


Figure 4: Les taux utilisés sont 0.75 (incubation) ; 0.05 (guérison) et 0.8 (transmission).

Application :

Les paramètres d'entrées :

- ✓N= 700
- ✓S0=699/700
- ✓E0=1/700
- ✓ I0=0/700
- ✓R0 = 0.0
- ✓γ = 0.07
- ✓ α = 0.14

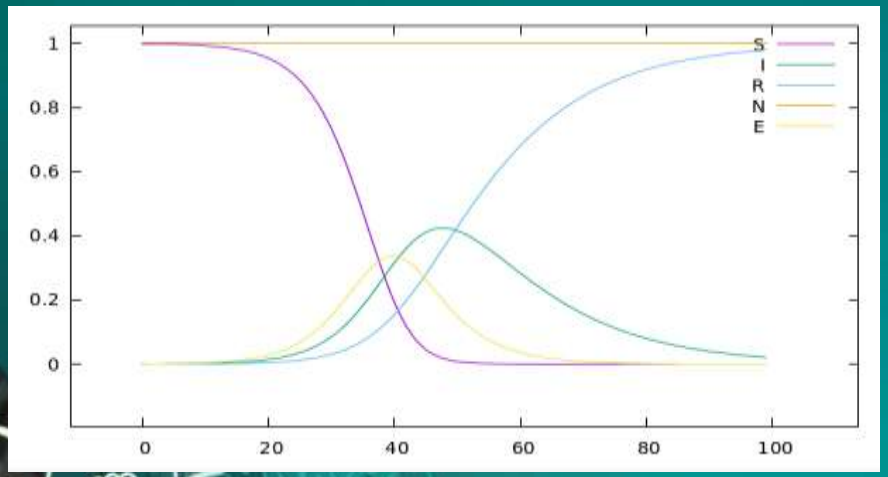


Figure 5 :le taux de transmission β =0,7
Mode d'enseignement : Présentiel

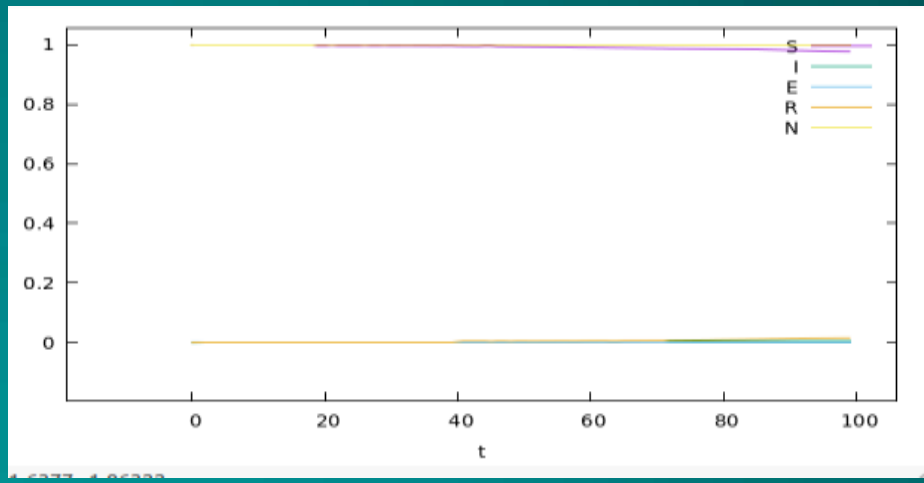


Figure 6 :le taux de transmission β =0,1.
Mode d'enseignement : En ligne

Modélisation de la cantine :

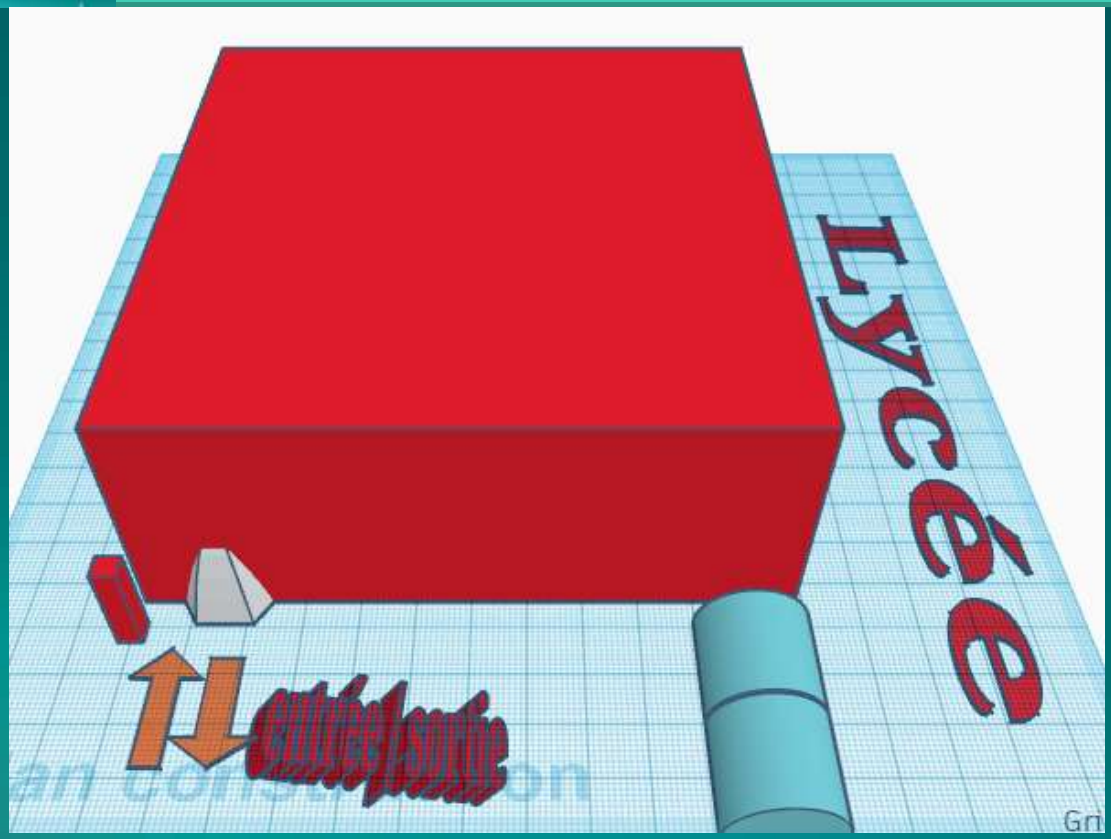


Figure 7 :modélisation 3D de la cantine (avec le programme tinkercad)

Principe Ultrasons :

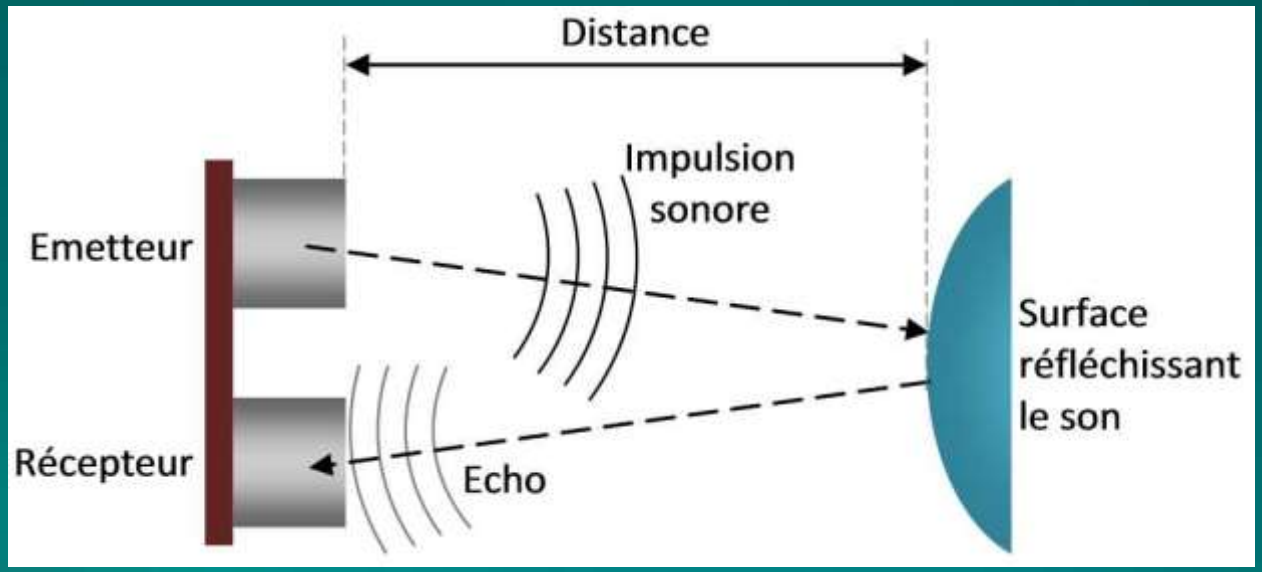


Figure 8 :schéma de fonctionnement de l'ultasons

Les composantes :

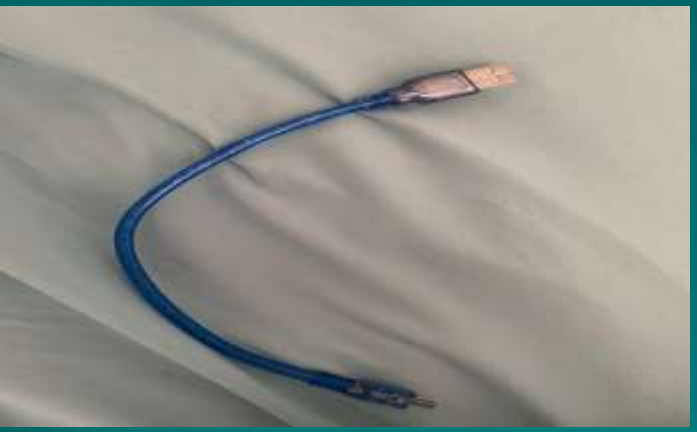


Figure 9 : Câble Arduino

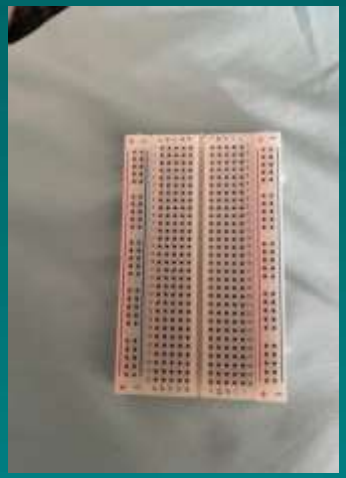


Figure 10: Breadboard

Figure 11 : Afficheur

Figure 12 : Buzzer

Les composantes :

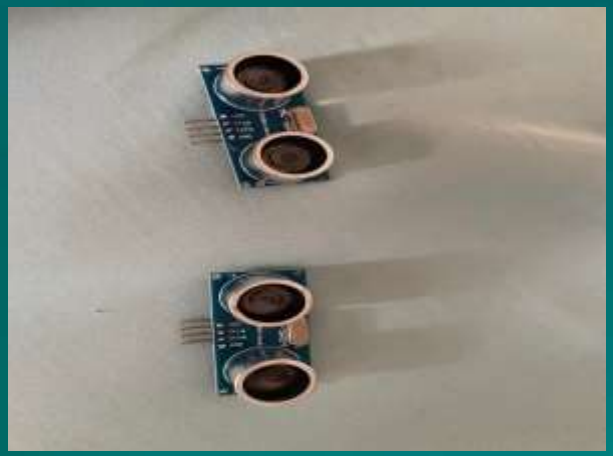


Figure 13 : Capteurs Ultrasons

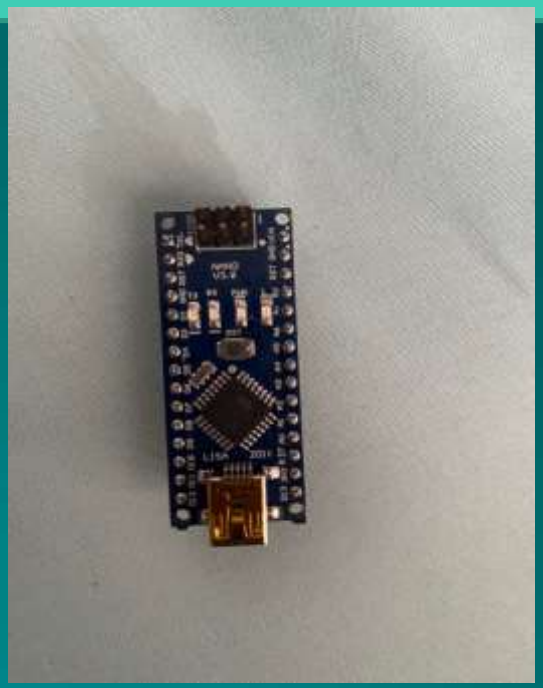


Figure 14 : Arduino Nano

Figure 15 : Câbles

Montage des composantes :

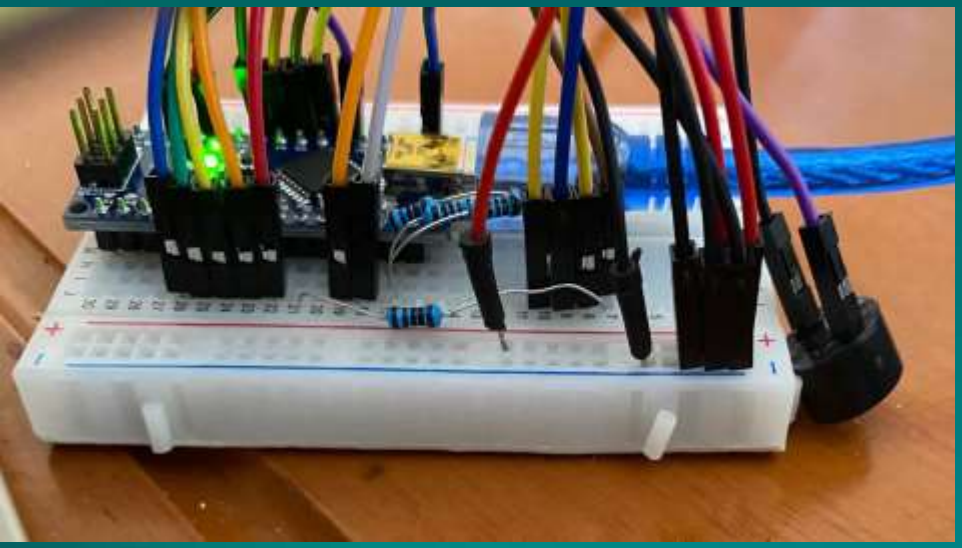


Figure 16 : Association du Buzzer avec la carte Arduino.

Figure 17: Association des capteurs Ultrasons avec Arduino

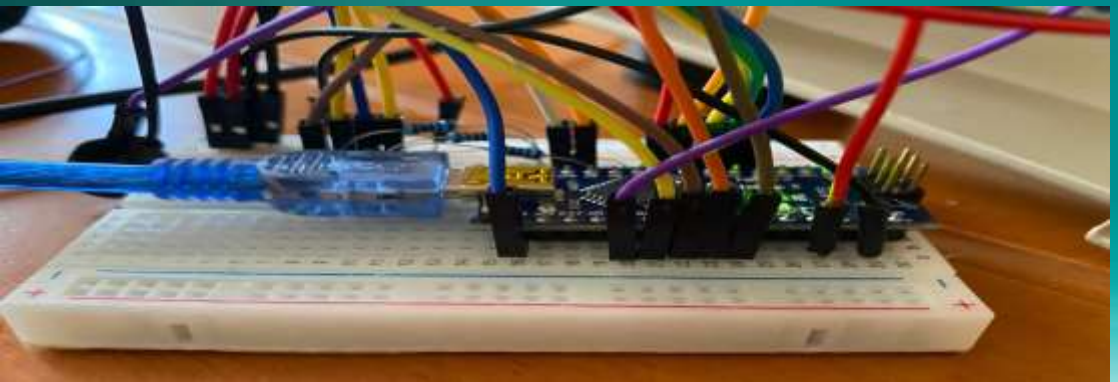


Figure 18 : Carte Arduino et les différents câbles qui la lie avec les autres composantes.

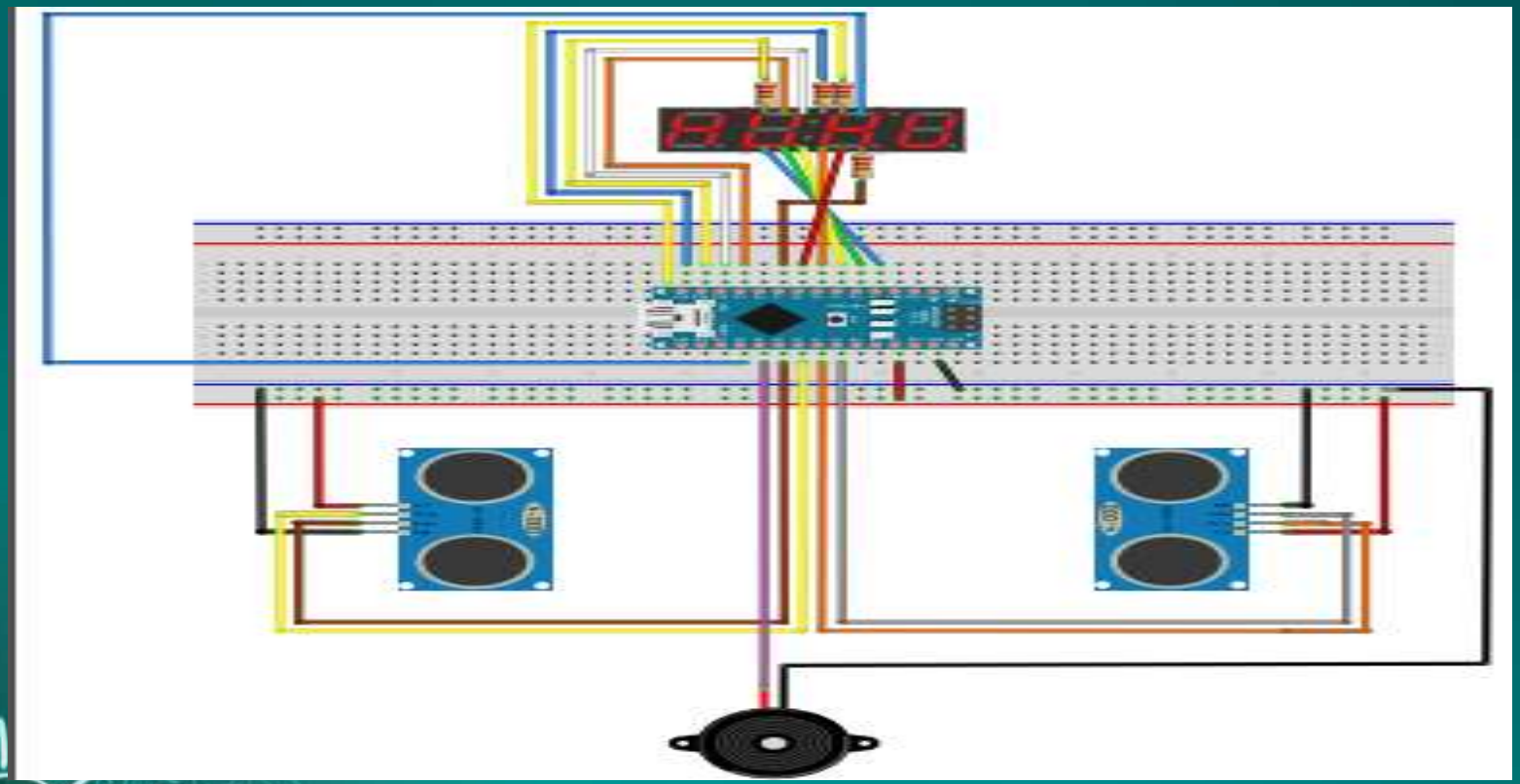


Figure 19 : Montage des composantes .

Application du programme informatique :

- *Figures qui résument le fonctionnement du programme informatique.*

Figure 20’

Figure 20

Figure 21.

Figure 21’ .

Figure 22.

Figure 22’.

Figure 23.

Figure 23’.

Figure 24.

Figure 24’.

Figure 25.

Figure 25’.

Conclusion :

I. **Modèles simples**
II. **Résultats raisonnables expliquant la réalité.**
III. **Résultats peuvent nous aider a prévoir le comportement du virus.**

MERCI POUR VOTRE ATTENTION

Annexe 1 :

```
1  import matplotlib.pyplot as plt
2
3  def SIR(beta,gamma,dt,tmax,s0,i0,r0):
4      nmax=int(tmax/dt)
5      S=[s0]
6      I=[i0]
7      R=[r0]
8      N=[S[0]+I[0]+R[0]]
9      t=[0]
10     #Euler explicite
11     for n in range (0, nmax-1):
12         S.append(S[n]-dt*beta*S[n]*I[n])
13         I.append(I[n]+dt*(beta*S[n]*I[n]-gamma*I[n]))
14         R.append(R[n]+dt*gamma*I[n])
15         N.append(S[n+1]+I[n+1]+R[n+1])
16         t.append((n+1)*dt)
17     plt.xlabel('temps (en jours)')
18     plt.ylabel('taux en % de la population')
19     plt.plot(t,S,label='S')
20     plt.plot(t,I,label='I')
21     plt.plot(t,R,label='R')
22     plt.plot(t,N,label='N')
23     plt.legend()
24     plt.show()
```

Annexe 2 :

```
1  import matplotlib.pyplot as plt
2
3  def SEIR(alpha,beta,gamma,dt,tmax,s0,e0,i0,r0):
4      nmax=int(tmax/dt)
5      S=[s0]
6      E=[e0]
7      I=[i0]
8      R=[r0]
9      N=[S[0]+E[0]+I[0]+R[0]]
10     t=[0]
11     #Euler explicite
12     for n in range (0, nmax-1):
13         S.append(S[n]-dt*beta*S[n]*I[n])
14         I.append(I[n]+dt*(alpha*E[n]-gamma*I[n]))
15         E.append(E[n]+dt*(beta*S[n]*I[n]-alpha*E[n]))
16         R.append(R[n]+dt*gamma*I[n])
17         N.append(S[n+1]+I[n+1]+E[n+1]+R[n+1])
18         t.append((n+1)*dt)
19     plt.xlabel('temps (en jours)')
20     plt.ylabel('taux en % de la population')
21     plt.plot(t,S,label='S')
22     plt.plot(t,E,label='E')
23     plt.plot(t,I,label='I')
24     plt.plot(t,R,label='R')
25     plt.plot(t,N,label='N')
26     plt.legend()
27     plt.show()
```

Annexe 3 :

adnane §

```
#include "SevSeg.h" // on utilise cette librairie pour faciliter l'affichage des donnés dans notre display 4_7 segments


SevSeg afficheur;

int maximum_personnes = 5; // le nombre de personnes maximum possible
int nb_personnes_maitenant = 0;

int alarme = 15; // le pin lié au + de alarme (alarme)
int capteur_ultra_son_1[] = {19,18}; //les pins liés au capteur utrasons 1
int capteur_ultra_son_2[] = {17,16}; // les pins liés au capteur utrasons 2
int capteur_ultra_son_1_Initial; // variable pour stocker la premiere distance mesuré par le premier capteur ;
//cette distance est la reference de detection de personnes aprés
int capteur_ultra_son_2_Initial; // variable pour stocker la premiere distance mesuré par le deuxieme capteur ;
//cette distance est la reference de detection de personnes aprés

String sequence = ""; //la sequence nous indique si un objet à ete detecé par nos capteur et la sequence cette
//detection exemple "12" vaut qu'un objet à eté detecté par le premier capteur apres il a ete detecte par le deuxieme

int timeoutCounter = 0;// variable utilisé pour reinitialiser la sequence si le temps entre la detection d'un capteur
//et l'attente de detection d'un autre capteur depasse un maximum


void setup() {
  //Setup code
  //Serial.begin(9600); // utilisé en cas de Serial.println() ; seulement dans l'etat de developement (pour le test et le deboggage)
  pinMode(alarme, OUTPUT);// on configure le pin lié à l'alarme (buzzer) dans l'etat output (sortie) pour pouvoir ecrire et forcer l'etat de ce pin
  delay(500);// on attend 500 ms et aprés on mesure les premiers distance qui seront la reference de nos detections aprés
```

Annexe 4 :

adnane §

```
capteur_ultra_son_1_Initial = mesureDistance(capteur_ultra_son_1);//le premier mesure de premier capteur ultrason
capteur_ultra_son_2_Initial = mesureDistance(capteur_ultra_son_2);//le premier mesure de deuxieme capteur ultrason


int type_4_7seg = COMMON_CATHODE; //le type de notre display 4_7 segments

//configuration des pins des nombres
int digit1 = 8; //Pin 8 lié a notre display correspond au premier nombre des 4 nombres
int digit2 = 11; //Pin 2 lié a notre display correspond au deuxieme nombre des 4 nombres
int digit3 = 12; //Pin 3 lié a notre display correspond au troisime nombre des 4 nombres
int digit4 = 7; //Pin 7 lié a notre display correspond au quatrieme nombre des 4 nombres

//configuration des pins des segments
int segA = 9; //Pin 9 lié a notre display correspond au segment A
int segB = 13; //Pin 13 lié a notre display correspond au segment B
int segC = 5; //Pin 5 lié a notre display correspond au segment C
int segD = 3; //Pin 3 lié a notre display correspond au segment D
int segE = 2; //Pin 2 lié a notre display correspond au segment E
int segF = 10; //Pin 10 lié a notre display correspond au segment F
int segG = 6; //Pin 6 lié a notre display correspond au segment G
int segDP= 4; //Pin 4 lié a notre display correspond au segment DP de la virgule

int nombre_des_digits = 4; // nombre des nombres de notre 4_7 segments
afficheur.Begin(type_4_7seg, nombre_des_digits, digit1, digit2, digit3, digit4, segA, segB, segC, segD, segE, segF, segG, segDP);//configurations et
//initialisation de notre display 4_7 segments
afficheur.SetBrightness(100); //configuration de la luminosité de notre display
```

Annexe 5 :

adnane §

```
}

void loop() {
  //lecture des distances basés sur les données recues de nos capteurs
  int capteur_ultra_son_1Val = mesureDistance(capteur_ultra_son_1);
  int capteur_ultra_son_2Val = mesureDistance(capteur_ultra_son_2);

  //Voir si un objet est detecté dans la zone capteur---------->capteur + capteur_ultra_son_1_Initial - 30 cm
  if(capteur_ultra_son_1Val < capteur_ultra_son_1_Initial - 30 && sequence.charAt(0) != '1'){
    sequence += "1"; //si la condition est satisfaite et la detection par le capteur ultra son 1 est faite pour la premier fois on ajoute 1 a la sequence
  }else if(capteur_ultra_son_2Val < capteur_ultra_son_2_Initial - 30 && sequence.charAt(0) != '2'){
    sequence += "2"; //si la condition est satisfaite et la detection par le capteur ultra son 2 est faite pour la premier fois on ajoute 1 a la sequence
  }

  if(sequence.equals("12")){ // si la sequence est egale  à "12" ce si est equivalent a deux detections succesives de capteur ultrason 1
  //après une detection du capteur ultrason 2
    nb_personnes_maitenant++;  // dans ce cas une personne est entré dans la salle donc on incremente le nombre de personnes
    sequence=""; // reinitialisation de la sequence
    delay(550);//550 // Stopper le programme pendant 550 ms pour qu'il n'y aura pas d'incrementation pour la meme personne
  }else if(sequence.equals("21") && nb_personnes_maitenant > 0){// si la sequence est egale  à "21" ce si est equivalent a deux detections
  //succesives de capteur ultrason 2 après une detection du capteur ultrason 1
    nb_personnes_maitenant--;  // dans ce cas une personne est sortie de la salle donc on decremente le nombre de personnes
    sequence=""; // reinitialisation de la sequence
    delay(550);// Stopper le programme pendant 550 ms pour qu'il n'y aura pas de decrementation pour la meme personne
  }
```

Annexe 6 :

adnane §

```
  }

  //reinisialisation du sequence si'elle est invalide ou bien on timeoutCounter atteint le maximum == 200
  if(sequence.length() > 2 || sequence.equals("11") || sequence.equals("22") || timeoutCounter > 200){
    sequence="";
  }

  // si un capteur a detecté une personne le timeoutCounter s'incremente
  if(sequence.length() == 1){ //
    timeoutCounter++;
  }else{
    timeoutCounter=0;
  }

  //si le nb de personnes attaint le maximum l'alarme est declenché
  if(nb_personnes_maitenant > maximum_personnes){
    tone(alarme, 1700);
  }else{
    noTone(alarme);
  }

  //affichage de nombre de personnes dans le 4_7 segments
  char tempString[10];
  sprintf(tempString, "%4d", nb_personnes_maitenant);
  afficheur.DisplayString(tempString,0);
}
```

Annexe 7 :

```
// Fonction qui prend en fonction les deux pins de capteur ultrason TRIG, ECHO pour mesurer la distance d'un capteur
int mesureDistance(int a[]) {
  pinMode(a[1], OUTPUT);//Configurer la broche Trig comme output (sortie)
  digitalWrite(a[1], LOW);// Éteignez la broche Trig au cas où elle était allumée et attendez 2 micro secondes.
  delayMicroseconds(2); //attendez 2 Micro seconds
  digitalWrite(a[1], HIGH);//Allumez et envoyez une onde sonore et attendez 10 micro secondes pour que l'opération se produise, puis éteignez la broche.
  delayMicroseconds(10);
  digitalWrite(a[1], LOW);//Éteignez la broche Trig
  pinMode(a[0], INPUT);//Configurer la broche Trig comme input (sortie)
  long duration = pulseIn(a[0], HIGH, 100000);//Obtenez le temps de réflexion des ondes sonores et convertissez la durée en centimètres
  return duration / 29 / 2;

  /*

  La vitesse de son: 340m/s = 29microseconds/cm
  on prend en compte la motié de distance parcourue par l'onde sonore reflichie par l'obstacle.
  DistanceInCms=microseconds/29/2
```