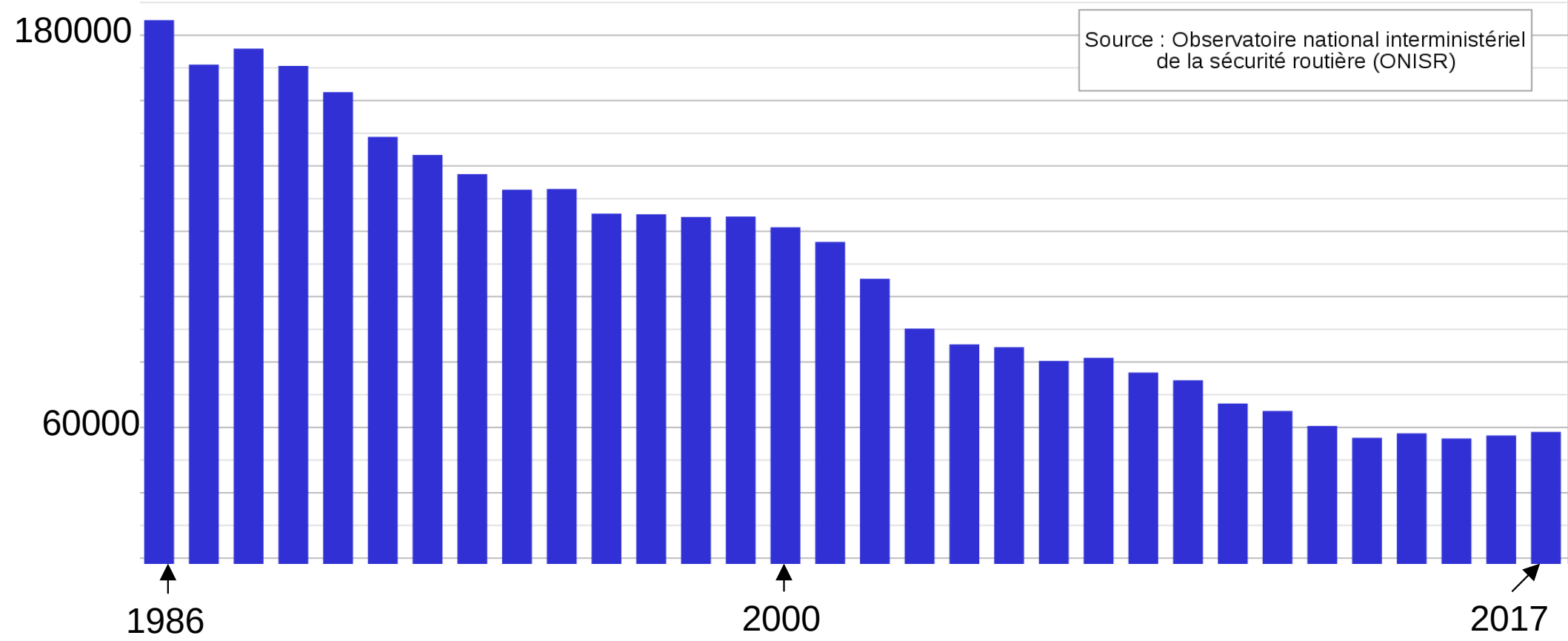
Correction automatique de trajectoire automobile

Parisel Guillaume

- Dans le monde : +1,35 M de morts sur les routes par an

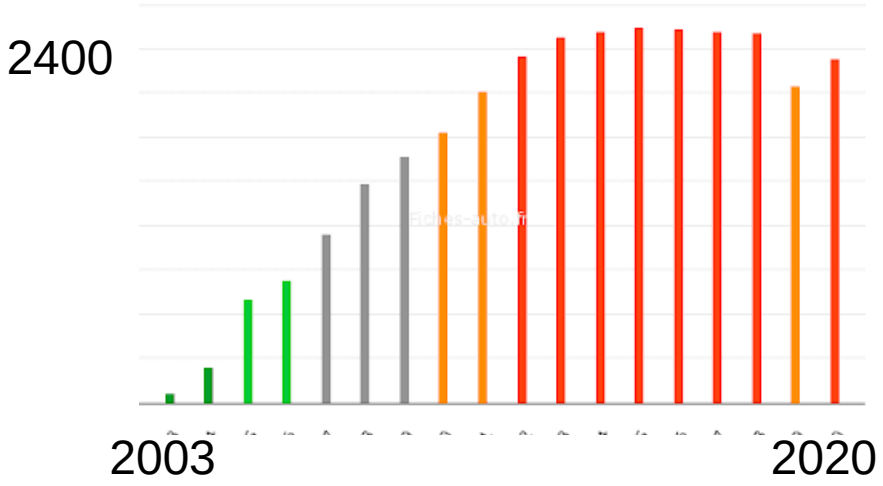
Nombres d’accidents de la route en france :



L’homme cherche à rendre la route plus sûre

Innovations sur la route

Nombre de radars automatiques en France



Innovations embarquées

Ceinture de sécurité obligatoire à l’avant :

1973

Et à l’arrière :

1990

- La technologie au service de la sécurité routière :

LKA (système de suivie de trajectoire)

Radars pédagogiques

AEB (freinage d’urgence)

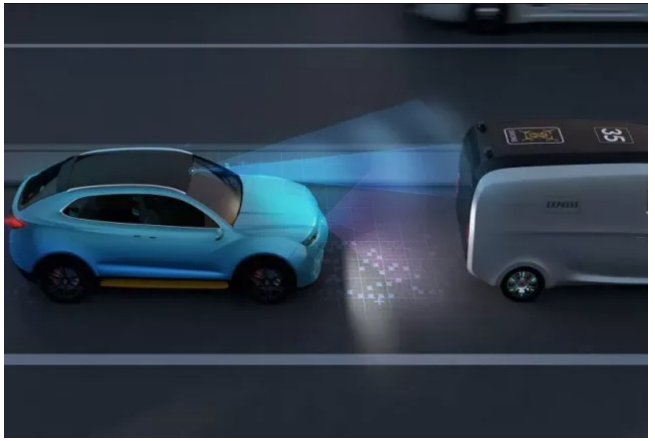


Image : avatacar.com, blog.stagespointspermis.com

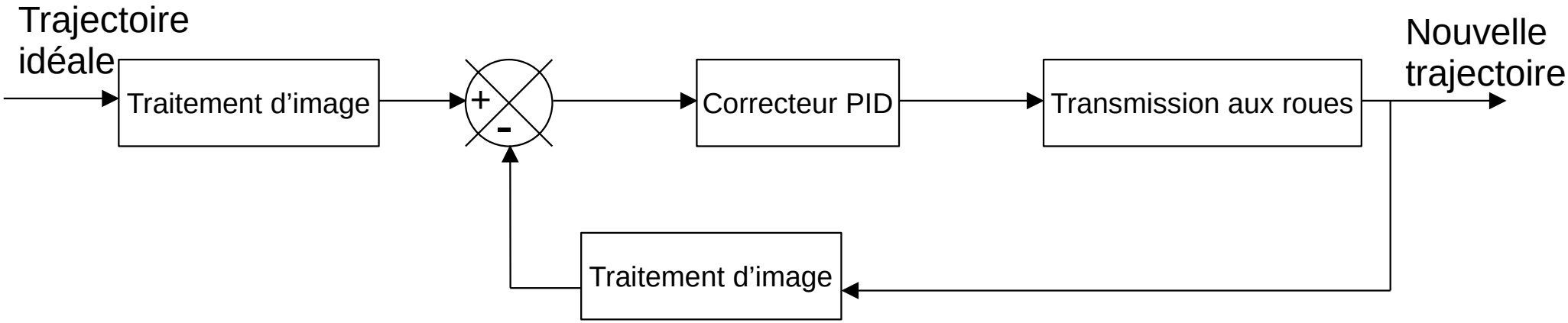
- Système adapté aux voies rapides
- Prédit la trajectoire et corrige :

Ligne bleu : prédiction

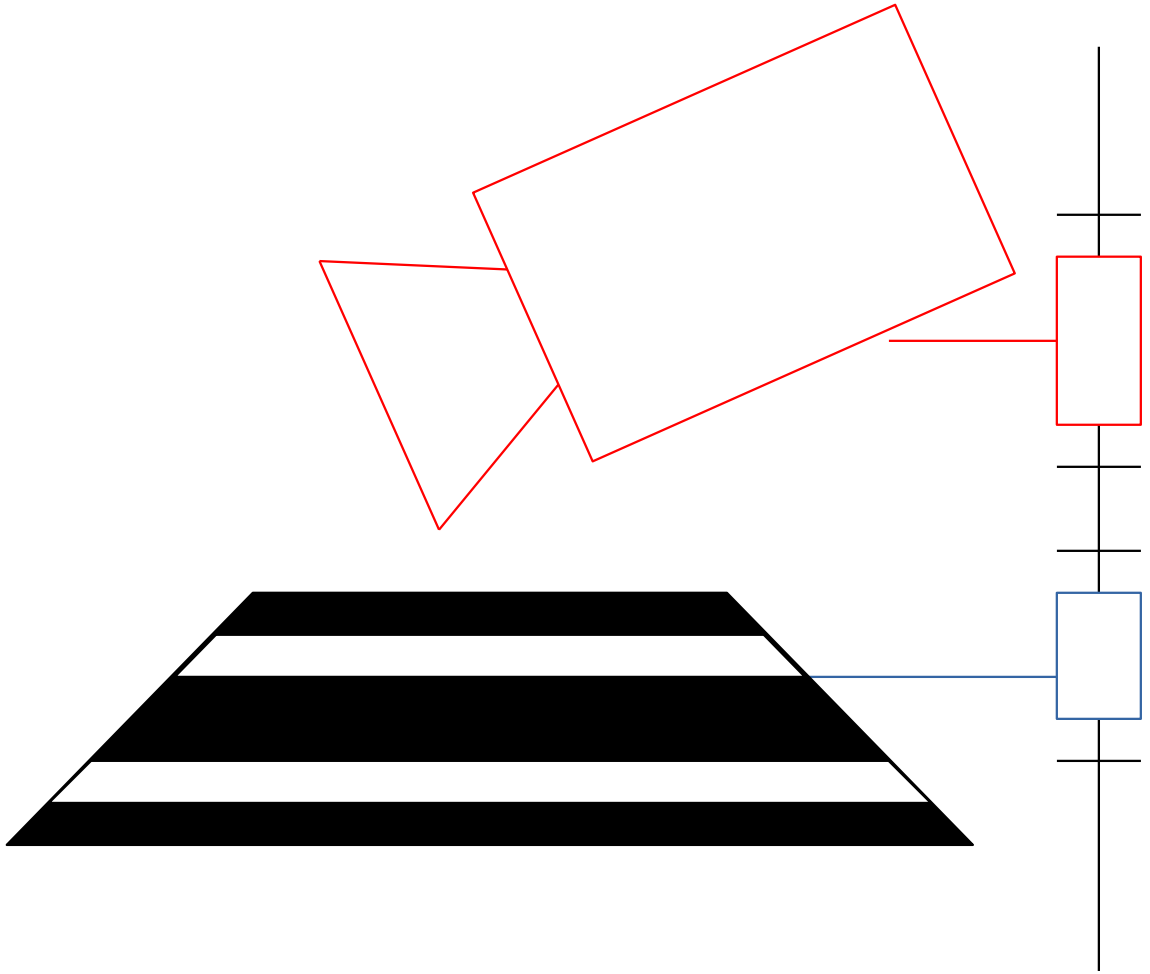
Lignes vertes : lignes de bords de route

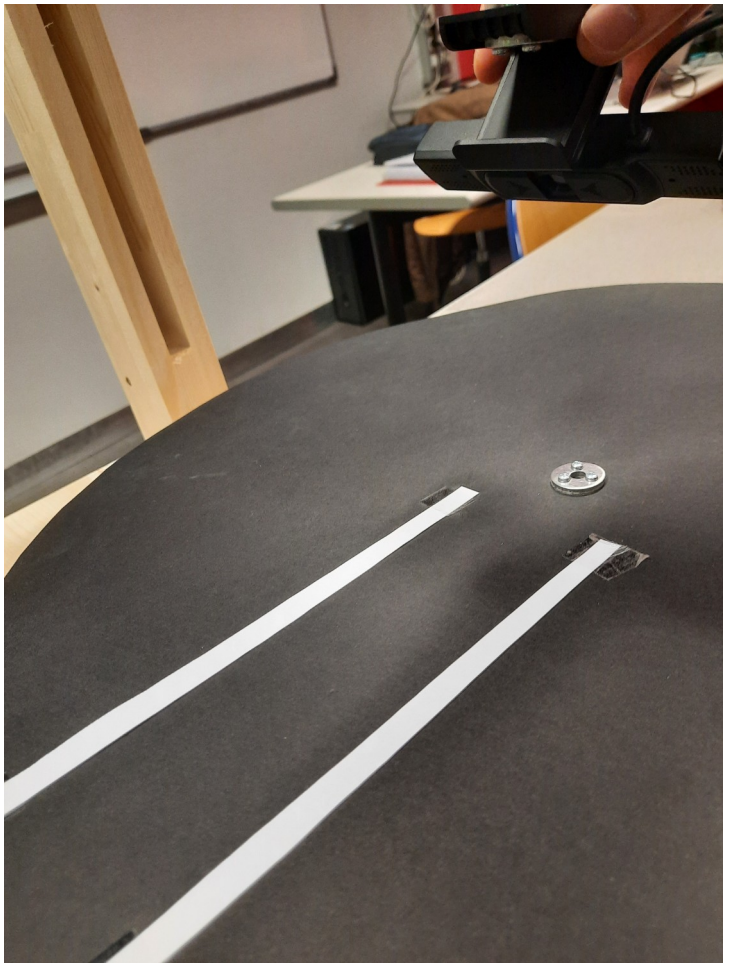
Étude de l’asservissement de l’orientation de roues

Schéma bloc du modèle réel :

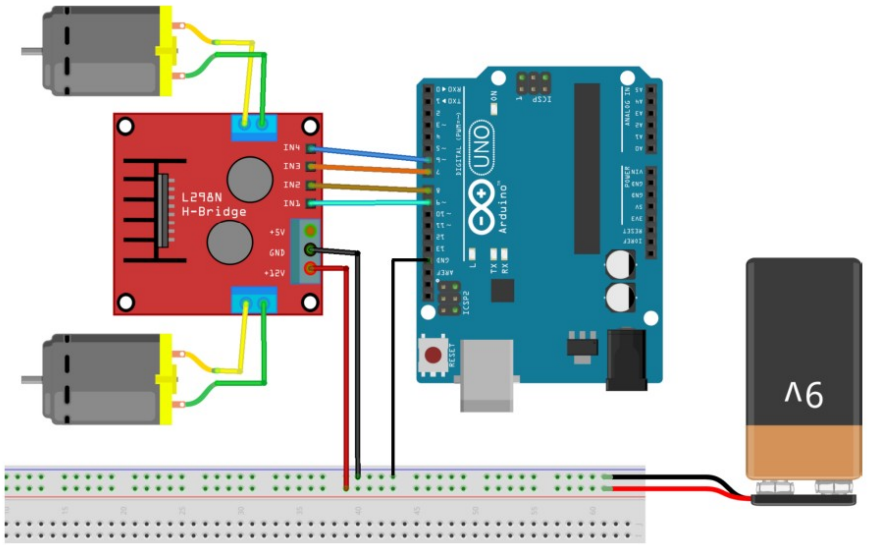


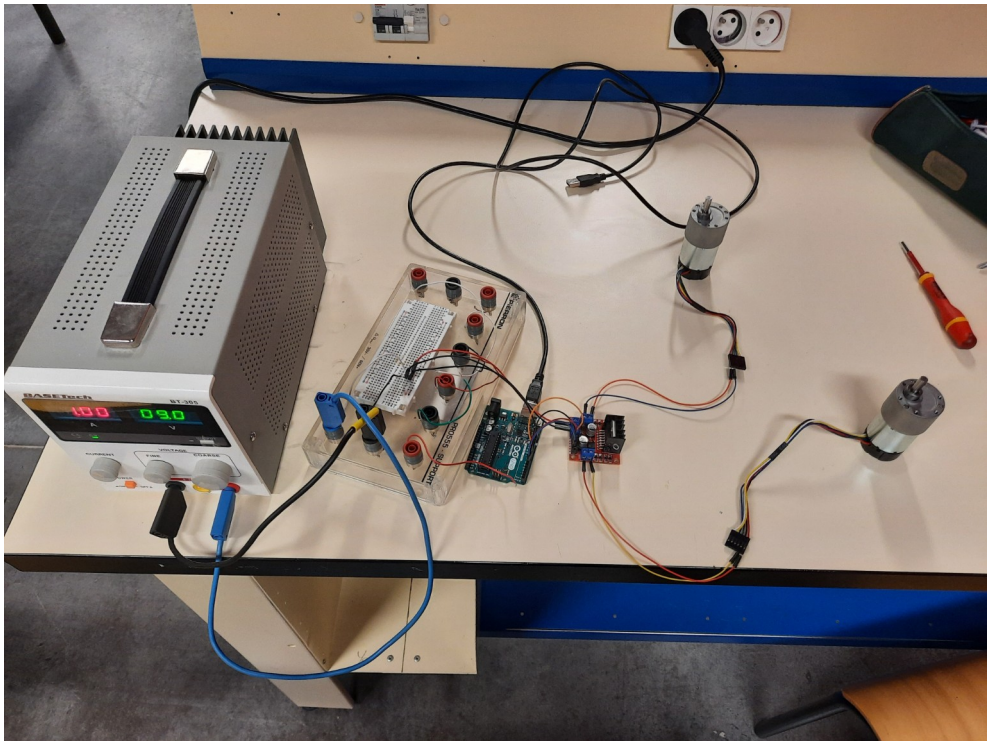
Modélisation :





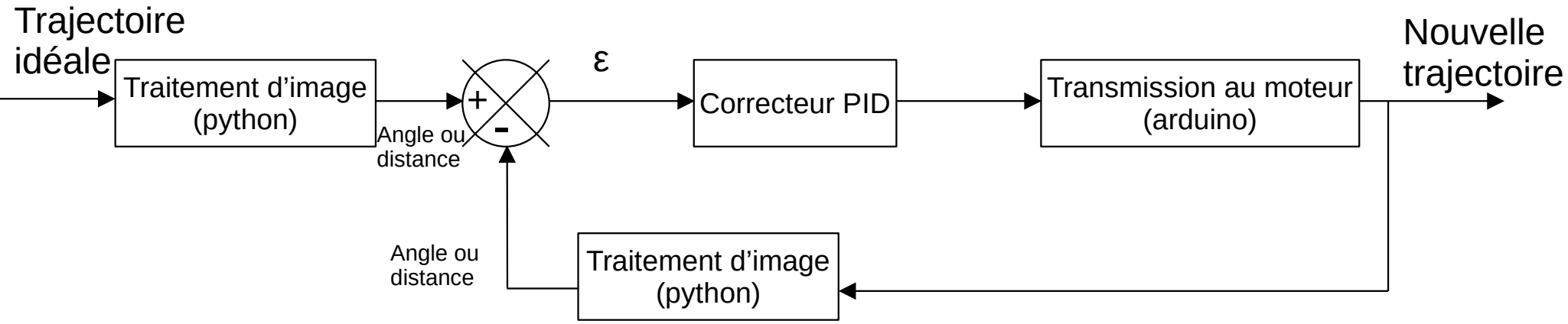
Modélisation





Commande des moteurs : arduino

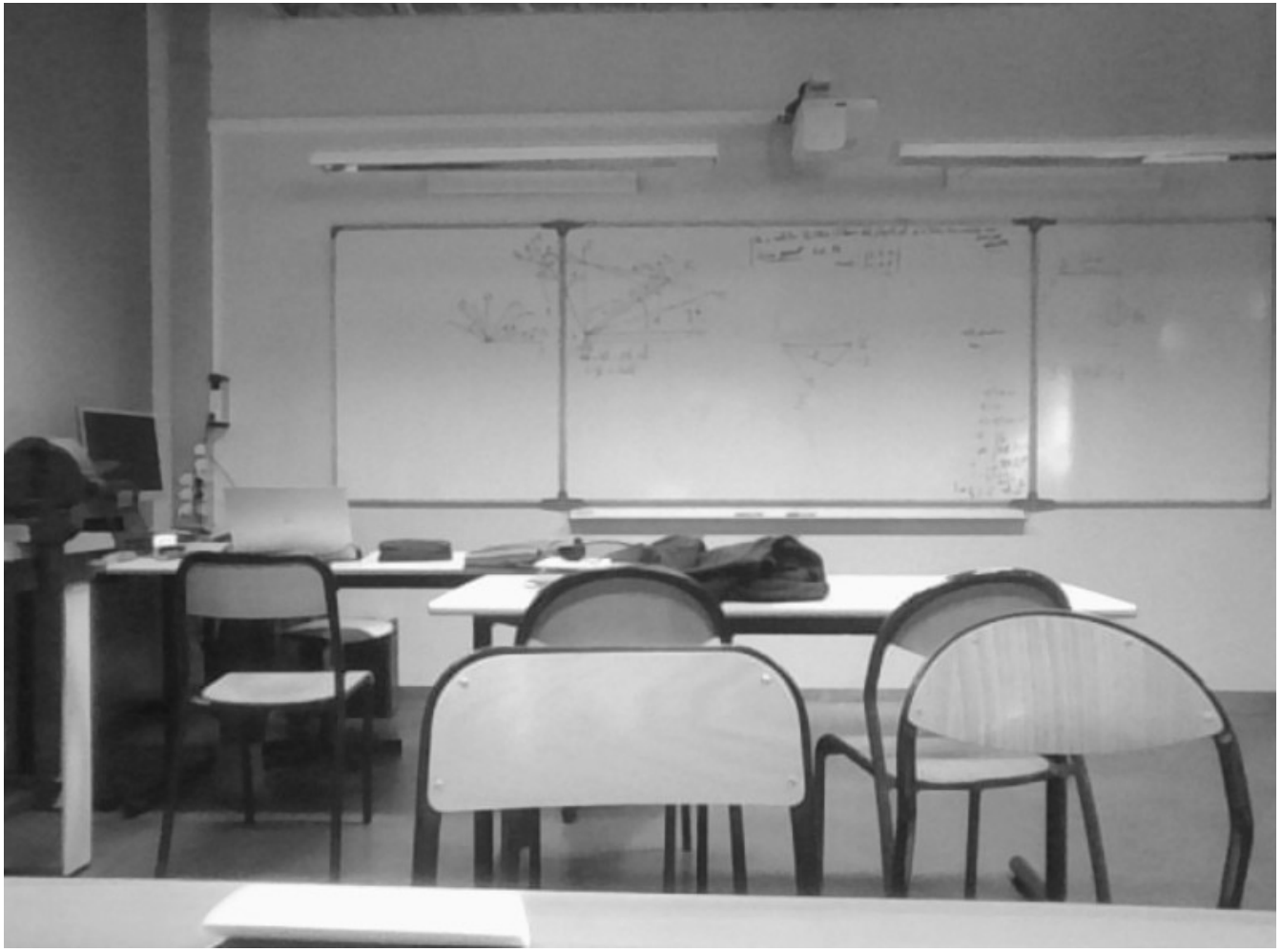
Schéma-bloc de ma modélisation



Traitement de l’image :

Contrainte : à faire en temps réel

Nuance de gris :



Moyenne du R, du V et du B

Noir et blanc (seuil) :

Seuil (Threshold en anglais) : comparer les pixels à un seuil et les « saturer »

Autre exemple : le filtre canny

Utilisation d'opencv

Code avec les outils du programme

```
import matplotlib.pyplot as plt
import matplotlib.image as pltim
import numpy as np
import time

seuilnb=0.5

img = pltim.imread('lena.png')

def niv_gris(image_test) :
    M, N, C = np.shape(image_test)
    g=[]
    for i in range (M) :
        d=[]
        for j in range (N):
            f=0
            for k in range (C):
                f+=image_test[i][j][k]
            d.append([f/3]*3)
        g.append(d)
    return g

def noir_blanc(image_test, val_seuil) :
    G = niv_gris(image_test)
    M, N, C = np.shape(image_test)
    for i in range(M) :
        for j in range(N) :
            if G[i][j][1] <= val_seuil :
                G[i][j] = [0]*3
            else :
                G[i][j] = [1.0]*3
    return G

t1=time.perf_counter()

image=noir_blanc(img,seuilnb)

t2=time.perf_counter()

print(t2-t1)

plt.imshow(image)
plt.show()
```

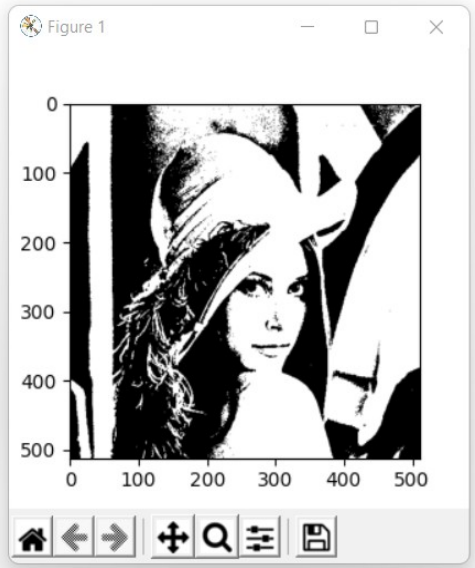
1,6 sec

Code opencv

```
import numpy as np
import cv2 as cv
import time

frame=cv.imread('lena.png')

s=127

t1=time.perf_counter()

gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)

s, black = cv.threshold(gray, s, 255, cv.THRESH_BINARY)

t2=time.perf_counter()

print(t2-t1)

cv.imshow('black and white', black)
cv.waitKey(0)

cv.destroyAllWindows()
```

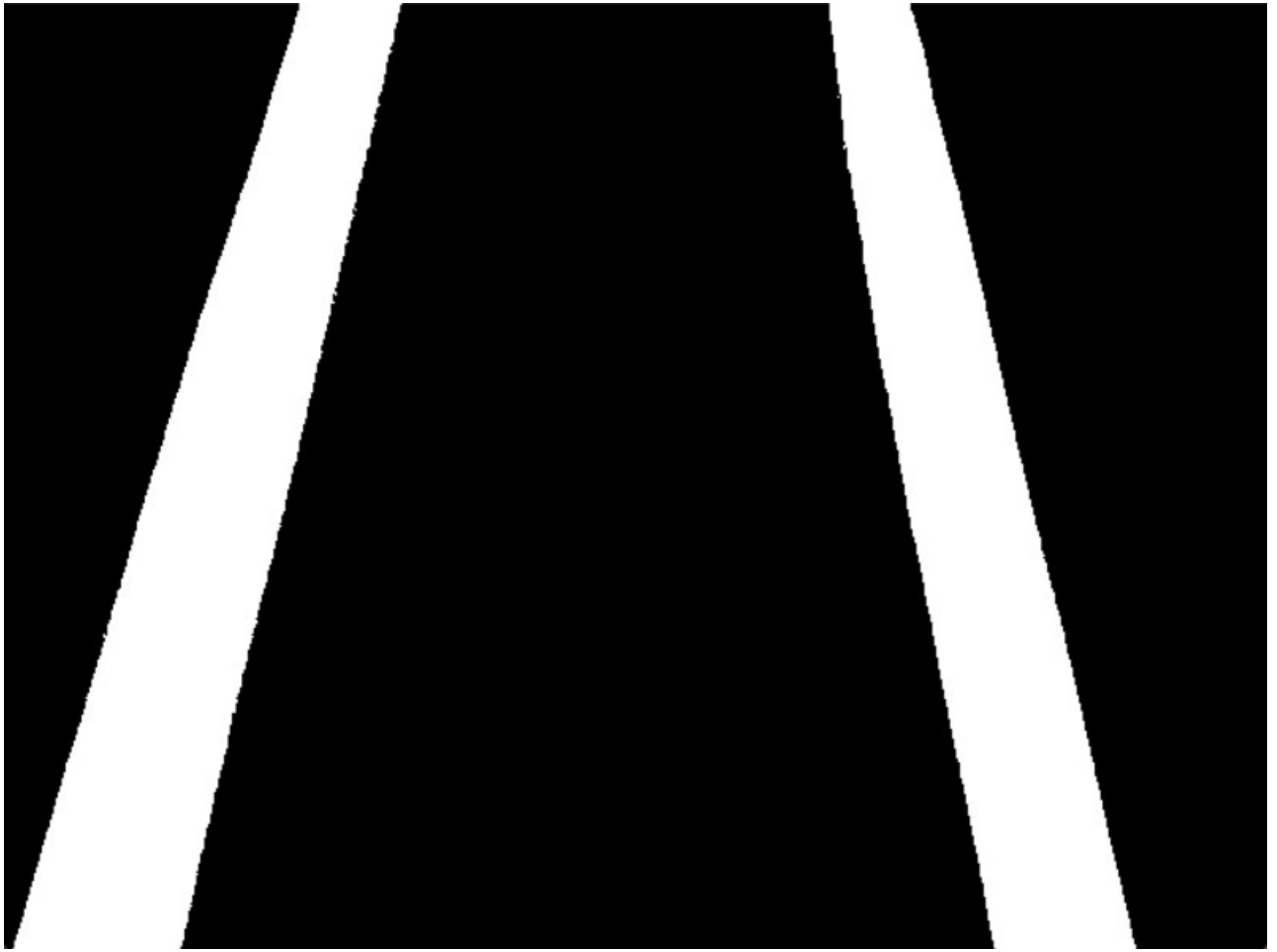
53 ms

```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: C:/Users/guill/OneDrive/Bureau/test temps open cv.py =======
0.0529006
>>>
========= RESTART: C:/Users/guill/OneDrive/Bureau/temps programme.py =========
1.5878589
```



30x plus rapide

Appliqué à la route :

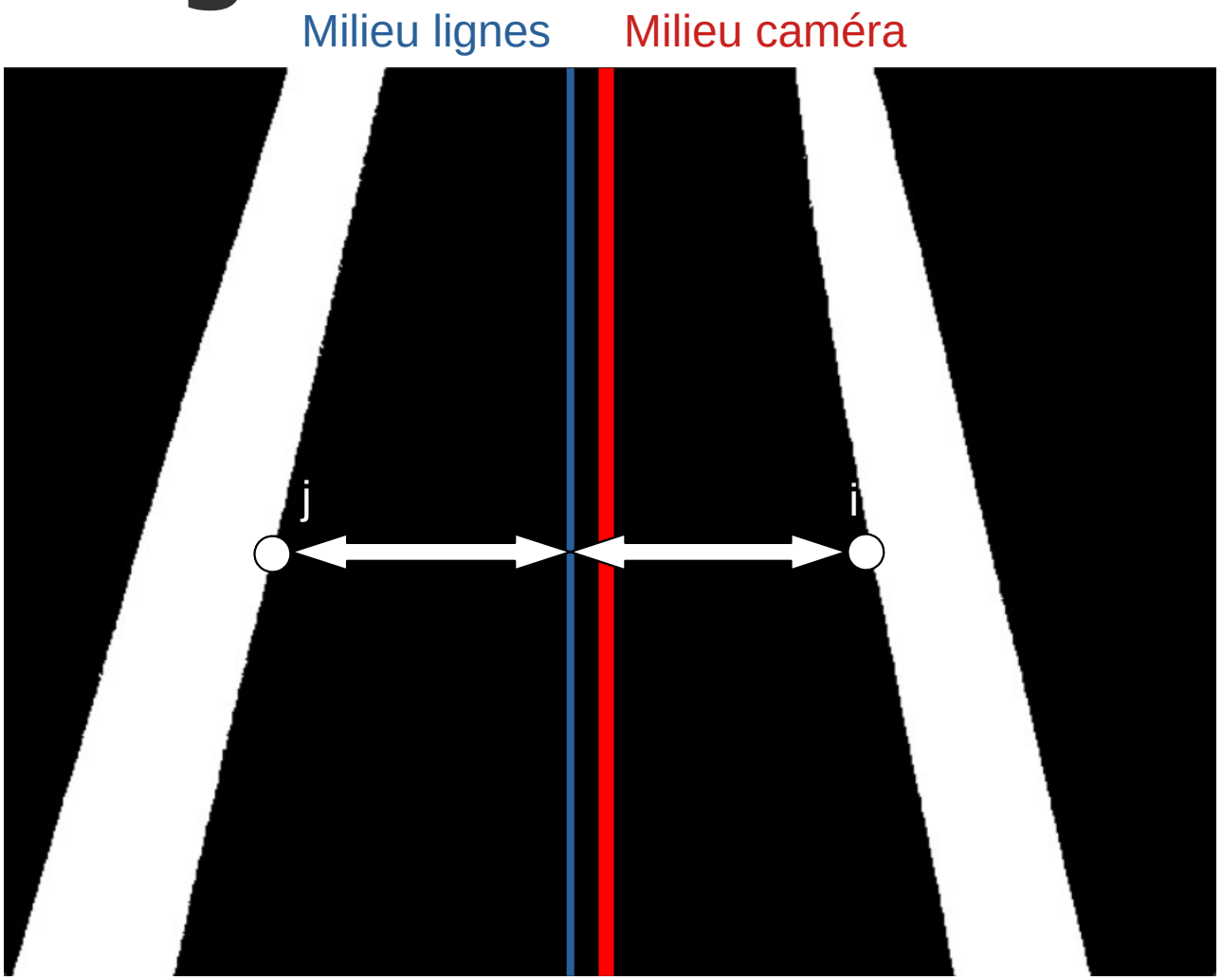


Comment définir la consigne?
Comment calculer l’erreur ?

Distance par rapport au milieu de l'image :

m : moitié de la largeur en pixels

i et j : abscisses de points



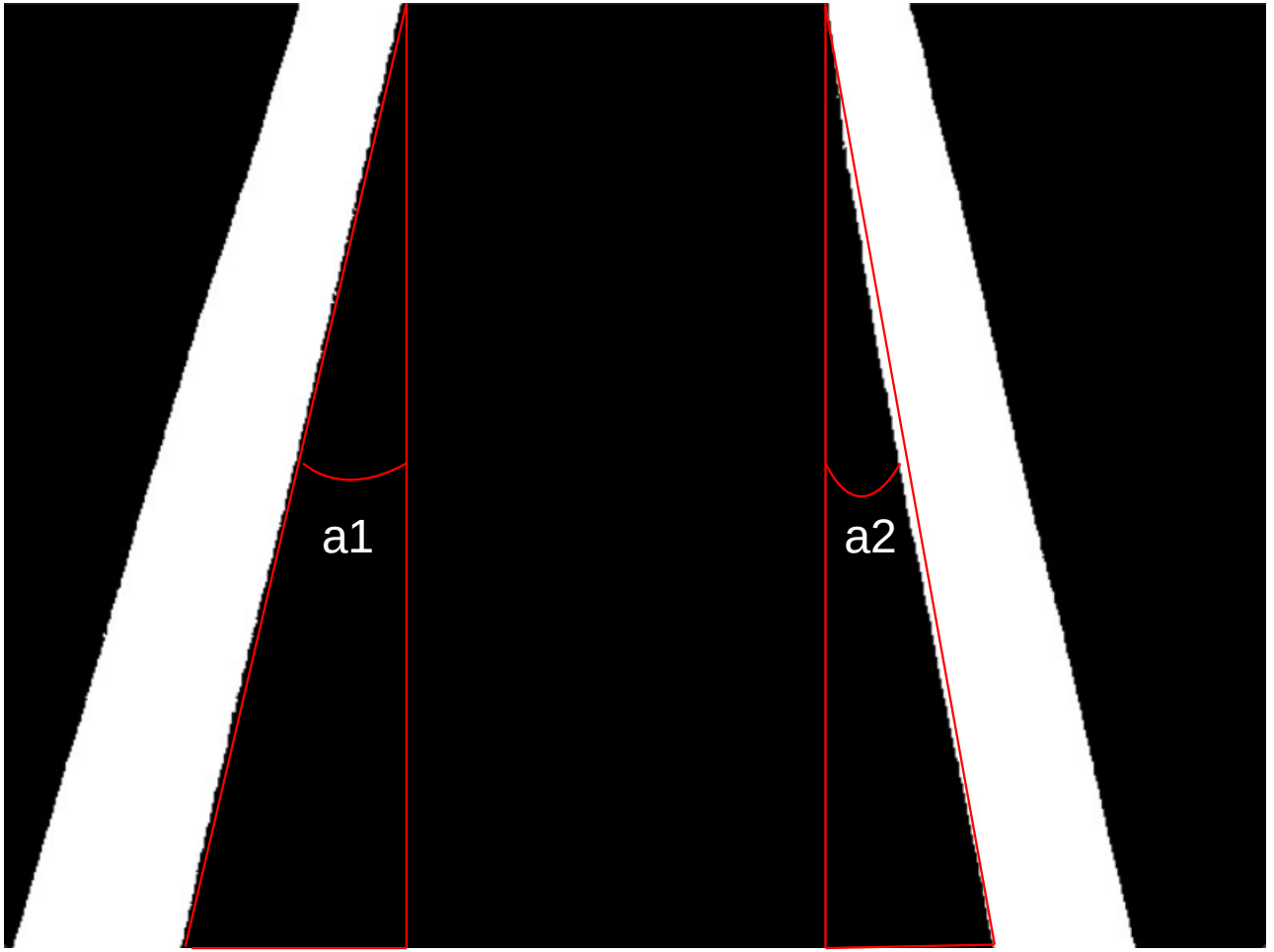
$$\frac{i+j}{2} - m$$

Cf annexe 1

Différence d’angles :

Abscisses des points du triangle

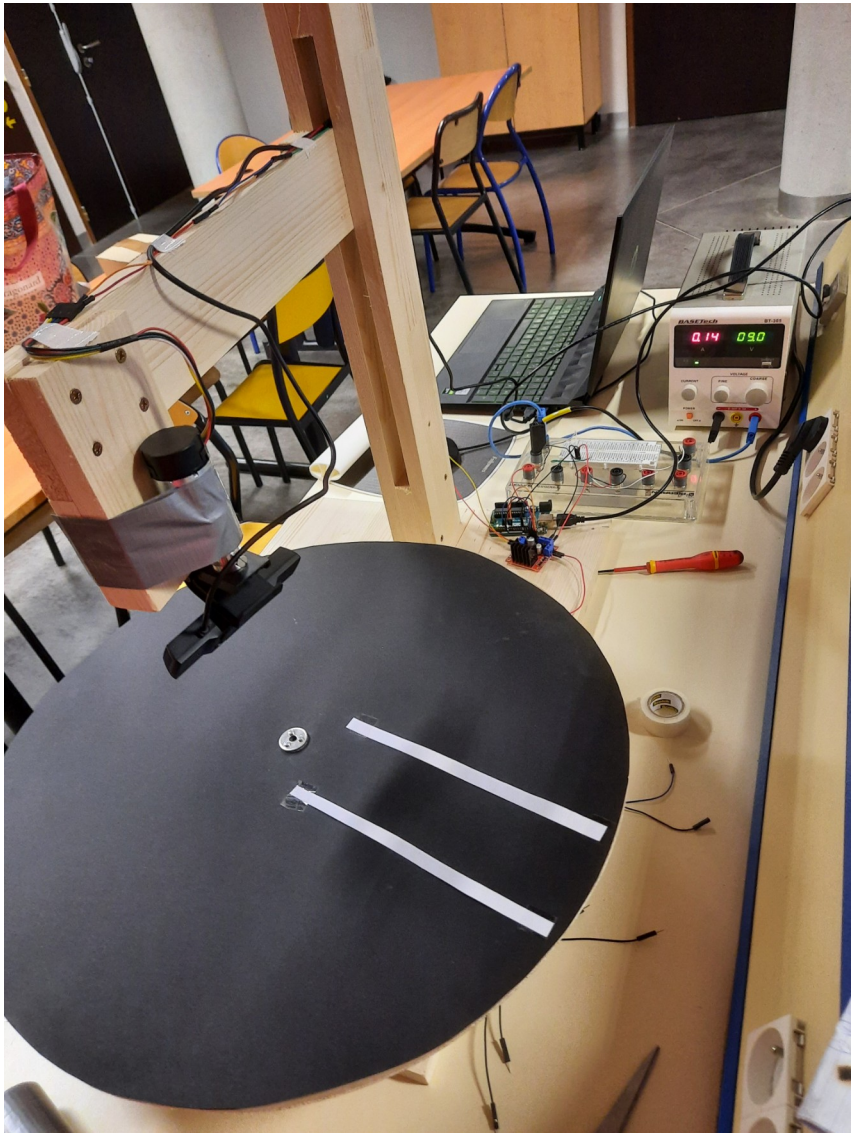
Ordonnée connue : hauteur image



Arctan pour trouver les angles

Cf annexe 2

Asservissement :



Problème de synchronisation arduino/python

Cf annexes

```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
cap = cv.VideoCapture(1)
if not cap.isOpened():
    print("Cannot open camera")
    exit()
ret0, frame0 = cap.read()
n, m = int(np.shape(frame0)[0]/2),int(np.shape(frame0)[1]/2)
s=170
t=0
T=[]
R=[]
def position_basique(M,N,mat):
    i=M
    j=M
    while i<2*M and mat[N,i] != 255 :
        i=i+1
    while j>0 and mat[N,j] != 255 :
        j=j-1
    return (i+j)/2-M
while True:
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    #cv.imshow('image', frame)
    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    #cv.imshow('gris', gray)
    s, black = cv.threshold(gray, s, 255, cv.THRESH_BINARY)
    cv.imshow('noir et blanc', black)
    #canny = cv.Canny(gray, 150, 175)
    #cv.imshow('canny', canny)
    r=position_basique(m,n,black)
    t=t+1
    T.append(t)
    R.append(r)
    if cv.waitKey(1) == ord('q'):
        break
plt.plot(T,R)
plt.show()
cap.release()
cv.destroyAllWindows()
```

Annexe 1

(Rapport au milieu de l’image)

Annexe 2

(différence d'angles)

```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
from math import atan
import serial
import serial.tools.list_ports

cap = cv.VideoCapture(0)

if not cap.isOpened():
    print("Cannot open camera")
    exit()

ret0, frame0 = cap.read()
n, m = int(np.shape(frame0)[0]/2),int(np.shape(frame0)[1]/2)
s=170
t=0
T=[]
R=[]
print("Recherche d'un port serie...")

ports = serial.tools.list_ports.comports(include_links=False)

if (len(ports) != 0): # on a trouvé au moins un port actif

    if (len(ports) > 1):     # affichage du nombre de ports trouvés
        print (str(len(ports)) + " ports actifs ont ete trouves:")
    else:
        print ("1 port actif a ete trouve:")

    ligne = 1

    for port in ports :  # affichage du nom de chaque port
        print(str(ligne) + ' : ' + port.device)
        ligne = ligne + 1


    baud = 9600

    # on établit la communication série

def position_extremites(M,N,mat):
    i=M
    j=M
    while i<2*M and mat[N,i] != 255 :
        i=i+1
    while j>0 and mat[N,j] != 255:
        j=j-1

    return j,i
```

Annexe 2 suite

```
while True:

    ret, frame = cap.read()

    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break

    #cv.imshow('image', frame)


    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)


    #cv.imshow('gris', gray)

    s, black = cv.threshold(gray, s, 255, cv.THRESH_BINARY)

    cv.imshow('noir et blanc', black)

    #canny = cv.Canny(gray, 150, 175)
    #cv.imshow('canny', canny)


    bas_gauche,bas_droit = position_extremites(m,0,black)
    haut_gauche,haut_droit = position_extremites(m,2*n-1,black)


    teta=(atan((haut_gauche-bas_gauche)/(2*n))+atan((haut_droit-bas_droit)/(2*n)))/2
    print(teta)
    t=t+1
    T.append(t)
    R.append(teta)

    vitesse=int(teta*850)

    if vitesse>255 :
        vitesse=255

    if vitesse<-255 :
        vitesse=-255

    message=str(vitesse)
    arr=bytes(message+"\r\n",'ascii')

    arduino = serial.Serial(ports[0].device, baud, timeout=1)
    arduino.write(arr)   # envoi du message série
    arduino.close()


        if cv.waitKey(1) == ord('q'):
            break

    plt.plot(T,R)
    plt.show()


    cap.release()
    cv.destroyAllWindows()
```

```
import matplotlib.pyplot as plt
import matplotlib.image as pltim
import numpy as np
import time
seuilnb=0.5
img = pltim.imread('lena.png')
def niv_gris(image_test) :
    M, N, C = np.shape(image_test)
    g=[]
    for i in range (M) :
        d=[]
        for j in range (N):
            f=0
            for k in range (C):
                f+=image_test[i][j][k]
            d.append([f/3]*3)
        g.append(d)
    return g
def noir_blanc(image_test, val_seuil) :
    G = niv_gris(image_test)
    M, N, C = np.shape(image_test)
    for i in range(M) :
        for j in range(N) :
            if G[i][j][1] <= val_seuil :
                G[i][j] = [0]*3
            else :
                G[i][j] = [1.0]*3
    return G
t1=time.perf_counter()
image=noir_blanc(img,seuilnb)
t2=time.perf_counter()
print(t2-t1)
plt.imshow(image)
plt.show()
```

Annexe 3

(Différence temps)

```
import numpy as np
import cv2 as cv
import time
frame=cv.imread('lena.png')
s=127
t1=time.perf_counter()
gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
s, black = cv.threshold(gray, s, 255, cv.THRESH_BINARY)
t2=time.perf_counter()
print(t2-t1)
cv.imshow('black and white', black)
cv.waitKey(0)
cv.destroyAllWindows()
```

Annexe 4

(Code arduino)

```
//Motor 1
const int motorPin1 = 9;
const int motorPin2 = 8;
//Motor 2
const int motorPin3 = 6;
const int motorPin4 = 5;

void setup(){
  //Set pins as outputs
  Serial.begin(9600);
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(motorPin3, OUTPUT);
  pinMode(motorPin4, OUTPUT);

}
void loop() {
  // put your main code here, to run repeatedly:
  int message=0;

  if (Serial.available() > 0) {
    message = Serial.readString().toInt();

    if (message > 0 ) {
      analogWrite(motorPin3, 0);
      analogWrite(motorPin4, message);
    }
    if (message < 0 ) {
      analogWrite(motorPin3, 0);
      analogWrite(motorPin4, -message);
    }
    if (message == 0 ) {
      analogWrite(motorPin3, 0);
      analogWrite(motorPin4, 0);
    }
  }
}
```