Blin Sébastien,
Collin Pierre-Henri,
Louarn Amaury

UNIVERSITÉ DE RENNES 1

# Université de Rennes 1

Campus de Beaulieu

## Licence STS

Cycle Préparatoire Ingénieur Rennes 1 - Informatique et Télécommunications

---

# Rapport de Travail d'Initiative Personnelle Encadrée (TIPE)

Comment la reconnaissance faciale du conducteur peut-elle améliorer sa sécurité au volant ?

Sous l'encadrement de :

Johanne Bézy-Wendling
Maître de Conférences
Responsable cycle préparatoire ingénieur de l'Université de Rennes I
(spécialité informatique et télécommunications)

Finn Jørgensen
Responsable L3 d'informatique - ISTIC

**Résumé**

# Introduction

### 0.0.1 Pourquoi ce projet

# Première partie

# Origine du problème

# Partie 1

# Historique de la reconnaissance faciale

# Partie 2

# Les enjeux de la sécurité routière

## Deuxième partie

# Reconnaissance Faciale

# Partie 3

# Théorie

## 3.1 Général

## 3.2 Eigenface

## 3.3 Fisherface

## 3.4 LBPH

# Partie 4

# Expérimentations

## 4.1 Protocole

## 4.2 Réalisation

## 4.3 Résultats et analyse

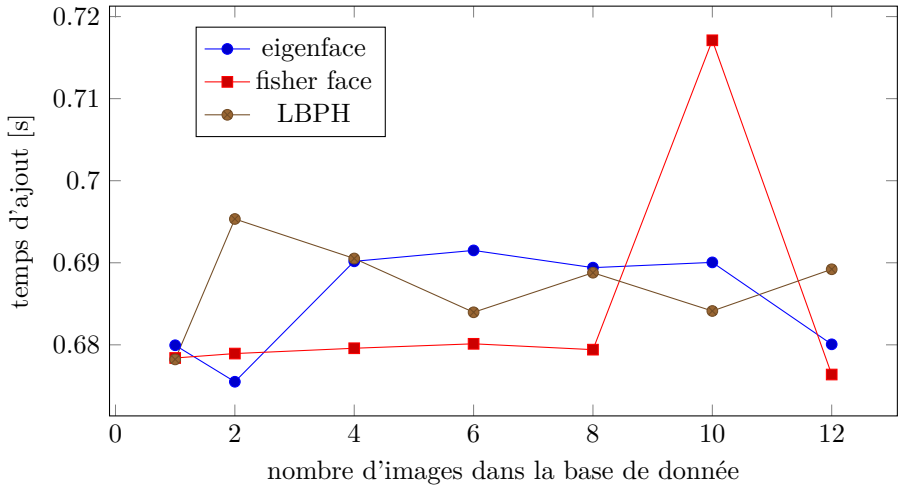


FIGURE 4.1 – Durée de reconnaissance par algorithme

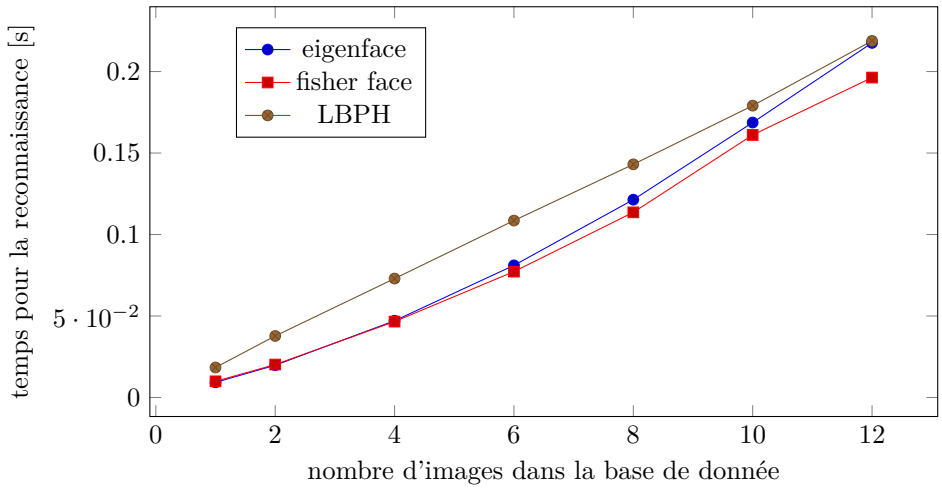


FIGURE 4.2 – Durée d'ajout des images dans la base de connaissance

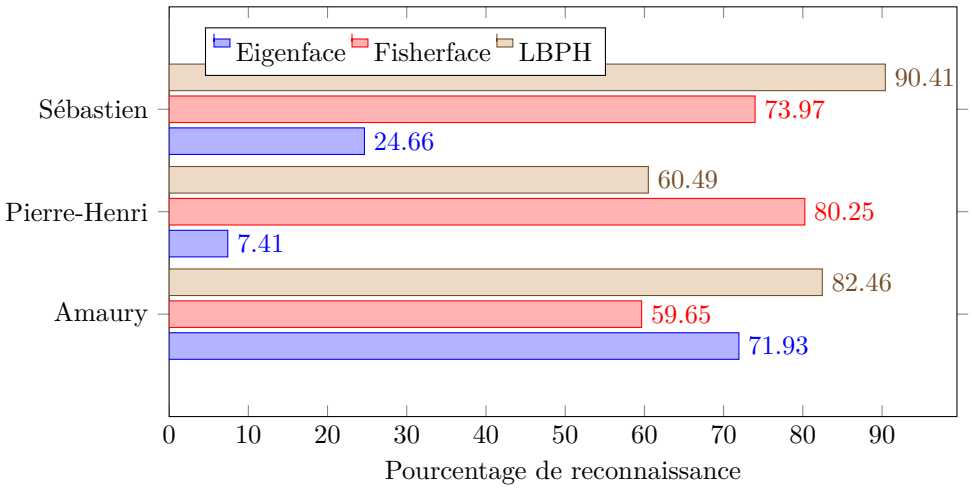


FIGURE 4.3 – Pourcentage de reconnaissance, en fonction du cobaye et de l'algorithme

# Troisième partie

# Reconnaissance des émotions

# Partie 5

# Théorie

## 5.1 Différentes solutions

## 5.2 Solution choisie

# Partie 6

# Expérimentations

## 6.1 Protocole

## 6.2 Réalisation

## 6.3 Résultats et analyse

# Quatrième partie

# Production

# Partie 7

# Prototype final

# Partie 8

# Tests finaux

# Partie 9

# Discussion et analyse

# Conclusion

# Annexe A

# Code des applications

## A.1 Application principale

```
1  #!/usr/bin/env python2
2  # -*- coding: utf-8 -*-
3
4  #   ooooooooooo ooooo ooooooooooo ooooooooooo       o       ooooooooooo   ooooooooo8
5  #   88  888  88  888    888     888 888             888       888     888 888
6  #       888      888    888ooooo88  888ooo8        8  88      888ooooo88   888ooooooo
7  #       888      888    888         888           8oooo88     888                  888
8  #      o888o    o888o o888o       o888ooo8888 o88o  o888o o888o        o88ooooo888
9
10 #Imports
11 import cv2
12 import numpy
13 import time
14 import os
15 import sys
16 import math
17 import serial
18 import glob
19 from collections import defaultdict
20
21 #Constantes
22 TRAINSET = "lbpcascade_frontalface.xml"     #Fichier de reconnaissance
23 IMAGE_SIZE = 170                            #Normalisation des images de base
24 NUMBER_OF_CAPTURE = 10                      #Nombre de captures a realiser pour la base de donnees
25 THRESHOLD = 90                                 #Seuil de reconnaissance
26 CAMERA = 1                                   #La camera
27 ARDUINO = False                             #Utiliser l'arduino ?
28
29 INDIVIDUS = []
30
31 ########################################
32 def sendSerial(ser, command):
33     """Envoie command a l'arduino"""
34     if(ARDUINO):
35         ser.write(command)
36
37 class CreateDataBase():
38     def __init__(self, imgPath ,ident):
39         self.rval = False
40         self.camera = cv2.VideoCapture(CAMERA)
```

```
41          self.classifier = cv2.CascadeClassifier(TRAINSET)
42          self.faceFrame = None
43          self.identity = ident
44          self.imagesPath = imgPath
45
46      def getFacesPos(self, frame):
47          """Retourne la position des visages detectes de la forme [[x y w h]]"""
48          faces = self.classifier.detectMultiScale(frame)
49          return faces
50
51      def drawDetectedFace(self, frame, faces):
52          """Dessine un rectangle autour du visage detecte"""
53          for f in faces:
54              x,y,w,h = [v for v in f]
55              cv2.rectangle(frame, (x,y), (x+w, y+h), (0,140,255))
56              self.LBPHBaseImage = self.getFaceFrame(frame, x, y, w, h)
57          return frame
58
59      def getFaceFrame(self, frame, x, y, w, h):
60          """On recupere un rectangle (largeur, hauteur) (centreX, centreY)"""
61          cropped = cv2.getRectSubPix(frame, (w, h), (x + w / 2, y + h / 2))
62          grayscale = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
63          self.faceFrame = cv2.resize(grayscale, (IMAGE_SIZE, IMAGE_SIZE))
64          return self.faceFrame
65
66      def collectFace(self, frame):
67          """On enregistre le visage recupere"""
68          imageCreated = False
69          captureNum = 0
70          #Cree le dossier s'il n'existe pas
71          try:
72              os.makedirs("{0}/{1}".format(self.imagesPath, self.identity))
73          except OSError:
74              print("ecriture dans dossier existant")
75          #Cree l'image a la suite
76          while not imageCreated:
77              if not os.path.isfile("{0}/{1}/{2}.jpg".format(self.imagesPath, self.identity, captureNu
78                  cv2.imwrite("{0}/{1}/{2}.jpg".format(self.imagesPath, self.identity, captureNum), fr
79                  imageCreated = True
80              else:
81                  captureNum += 1
82
83      def capture(self):
84          """Recupere le flux video"""
85          if self.camera.isOpened():
86              (rval, frame) = self.camera.read()
87          else:
88              rval = False
89
90          while rval:
91              (rval, frame) = self.camera.read()
92              frame = self.drawDetectedFace(frame, self.getFacesPos(frame))
93              #Affichage du texte
94              cv2.putText(frame, "Appuyez sur c pour collecter", (0,20), cv2.FONT_HERSHEY_SIMPLEX, 0.8
95              cv2.imshow("Creation de la BDD", frame)
96              key = cv2.waitKey(20)
97              if key in [27, ord('Q'), ord('q')]: #esc / Q
98                  break
99              if key in [ord('C'), ord('c')] and self.faceFrame != None:
100                 self.collectFace(self.faceFrame)
101
102 class Recognize():
```

```
103     def __init__(self, imgPath):
104         self.rval = False
105         self.camera = cv2.VideoCapture(CAMERA)
106         self.classifier = cv2.CascadeClassifier(TRAINSET)
107         self.faceFrame = None
108         self.identities = []
109         self.imagesPath = imgPath
110         self.images = []
111         self.imagesIndex = []
112         self.time = time.time()
113 
114         self.eyeWide = 0
115         self.eyeHeight = 0
116         self.grayMouthClosed = 0
117         self.thresholdEyeClosed = 0
118 
119     def getFacesPos(self, frame):
120         """Retourne la position des visages detectes de la forme [[x y w h]]"""
121         faces = self.classifier.detectMultiScale(frame)
122         return faces
123 
124     def drawDetected(self, frame, detected, color):
125         """Dessine un rectangle autour du visage detecte"""
126         if detected is None:
127             return frame
128         for d in detected:
129             x,y,w,h = [v for v in d]
130             cv2.rectangle(frame, (x,y), (x+w, y+h), color)
131         return frame
132 
133     def getFaceFrame(self, frame, x, y, w, h):
134         """On recupere un rectangle (largeur, hauteur) (centreX, centreY)"""
135         cropped = cv2.getRectSubPix(frame, (w, h), (x + w / 2, y + h / 2))
136         grayscale = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
137         self.faceFrame = cv2.resize(grayscale, (IMAGE_SIZE, IMAGE_SIZE))
138         return self.faceFrame
139 
140     def getCroppedEyesPos(self, croppedFrame):
141         """Retourne la position des bouches detectes de la forme [[x y w h]]"""
142         cascade = cv2.CascadeClassifier('haarcascade_lefteye_2splits.xml')
143         rects = cascade.detectMultiScale(croppedFrame)
144         if len(rects) == 0:
145             return rects
146         final = None
147         x1 = 0
148         x2 = 0 + len(croppedFrame)
149         y1 = 0
150         y2 = 0 + len(croppedFrame[0])*1/2
151 
152         #Prend la partie inferieure de la tete pour le traitement
153         for rect in rects:
154             if rect[0] > x1 and rect[0] + rect[2] < x2 and rect[1] > y1 and rect[1] + rect[3] < y2:
155                 if final is None:
156                     final = [rect]
157                 else:
158                     final += [rect]
159         return final
160 
161     def getCroppedMouthPos(self, croppedFrame):
162         """Retourne la position des bouches detectes de la forme [[x y w h]]"""
163         cascade = cv2.CascadeClassifier('mouth_classifier.xml')
164         rects = cascade.detectMultiScale(croppedFrame)
```

```
165             if len(rects) == 0:
166                 return rects
167             final = None
168             x1 = 0
169             x2 = 0 + len(croppedFrame)
170             y1 = 0 + len(croppedFrame[0])*5/8
171             y2 = 0 + len(croppedFrame[0])
172 
173             #Prend la partie inferieure de la tete pour le traitement
174             for rect in rects:
175                 if rect[0] > x1 and rect[0] + rect[2] < x2 and rect[1] > y1 and rect[1] + rect[3] < y2:
176                     if final is None:
177                         final = [rect]
178                     else:
179                         final += [rect]
180             return final
181 
182         def extractAndResize(self, frame, x, y, w, h):
183             """On recupere juste la tete en noir et blanc"""
184             cropped = cv2.getRectSubPix(frame, (w, h), (x + w / 2, y + h / 2))
185             grayscale = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
186             resized = cv2.resize(grayscale, (IMAGE_SIZE, IMAGE_SIZE))
187             return resized
188 
189         def cropFromFace(self, frame, facePos):
190             """Garde seulement la partie "tete" de la frame"""
191             #X,Y,W,H
192             if facePos is None:
193                 return frame
194             if len(facePos) == 0 :
195                 return frame
196             else :
197                 x1 = facePos[0][0]
198                 x2 = x1 + facePos[0][2]
199                 y1 = facePos[0][1]
200                 y2 = y1 + facePos[0][3]
201                 return frame[y1:y2, x1:x2]
202 
203         def readImages(self):
204             """Recupere les images de bases pour effectuer la reconnaissance des visages"""
205             c = 0
206             self.images = []
207             self.imagesIndex = []
208             for dirname, dirnames, filenames in os.walk(self.imagesPath):
209                 for subdirname in dirnames:
210                     self.identities.append(subdirname)
211                     subject_path = os.path.join(dirname, subdirname)
212                     for filename in os.listdir(subject_path):
213                         try:
214                             im = cv2.imread(os.path.join(subject_path, filename), 0)
215                             self.images.append(numpy.asarray(im, dtype=numpy.uint8))
216                             self.imagesIndex.append(c)
217                         except IOError, (errno, strerror):
218                             print "I/O error({0}): {1}".format(errno, strerror)
219                         except:
220                             print "Unexpected error:", sys.exc_info()[0]
221                             raise
222                     c += 1
223 
224         def recognizeLBPHFace(self):
225             """Reconnait par la methode LBPH"""
226             self.model = cv2.createLBPHFaceRecognizer()
```

```
227             self.model.train(numpy.asarray(self.images), numpy.asarray(self.imagesIndex))
228
229         def recognize(self):
230             """On choisit la methode de reconnaissance et on construit la base de donnee"""
231             self.readImages()
232             self.recognizeLBPHFace()
233             if not self.camera.isOpened():
234                 return
235             self.capture()
236
237         def identify(self, image):
238             """On reconnait l'identite de la personne si enregistree"""
239             [p_index, p_confidence] = self.model.predict(image)
240             found_identity = self.identities[p_index]
241             return found_identity, p_confidence
242
243         def initNeutral(self, neutralImg):
244             """Initialise les thresholds + les largeurs/hauteurs pour la detection des emotions"""
245             frame = neutralImg
246             facePos = self.getFacesPos(frame)
247             cropped = self.cropFromFace(frame, facePos)
248             mouthPos = self.getCroppedMouthPos(cropped)
249             mouthFrame = self.cropFromFace(frame, mouthPos)
250             gray = cv2.cvtColor(mouthFrame, cv2.COLOR_BGR2GRAY)
251             ret,thresh = cv2.threshold(gray,50,255,cv2.THRESH_BINARY)
252             self.grayMouthClosed = numpy.count_nonzero(thresh)
253             eyePos = self.getCroppedEyesPos(cropped)
254             eyeFrame = self.cropFromFace(frame, eyePos)
255             hsv = cv2.cvtColor(eyeFrame, cv2.COLOR_BGR2HSV)
256             lowerColor = numpy.array([80, 0,0])
257             upperColor = numpy.array([160,100,100])
258             mask = cv2.inRange(hsv, lowerColor, upperColor)
259             self.thresholdEyeClosed = numpy.count_nonzero(mask)
260             self.eyeWide = eyePos[0][2]
261             self.eyeHeight = eyePos[0][3]
262             return 0
263
264
265         def isMouthOpen(self, mouthFrame):
266             gray = cv2.cvtColor(mouthFrame, cv2.COLOR_BGR2GRAY)
267             ret,thresh = cv2.threshold(gray,50,255,cv2.THRESH_BINARY)
268             return self.grayMouthClosed < numpy.count_nonzero(thresh)-300
269
270
271         def EyeNotHeightThanNeutral(self, EyeHeight, threshold):
272             """Determine si les yeux sont fronces"""
273             return EyeHeight < self.eyeHeight - threshold
274
275         def EyeHeightThanNeutral(self, EyeHeight, threshold):
276             """Determine si les yeux sont equarquilles"""
277             return EyeHeight > self.eyeHeight + threshold
278
279         def isEyeClosed(self, eyeFrame):
280             hsv = cv2.cvtColor(eyeFrame, cv2.COLOR_BGR2HSV)
281             lowerColor = numpy.array([80, 0,0])
282             upperColor = numpy.array([160,100,100])
283             mask = cv2.inRange(hsv, lowerColor, upperColor)
284             return self.thresholdEyeClosed*2 < numpy.count_nonzero(mask)
285
286
287         def emotions(self):
288             """Recupere le flux video"""
```

```
289             interval = 0
290             dontlook = 0
291             sleep = 0
292             mouthOpen = 0
293             eyeClose = 0
294             eyeBigger = 0
295             eyeNotBigger = 0
296             error = 0
297             if self.camera.isOpened():
298                 (rval, frame) = self.camera.read()
299             else:
300                 rval = False
301             i = 0
302             while rval:
303                 (rval, frame) = self.camera.read()
304                 facePos = self.getFacesPos(frame)
305                 if len(facePos) is 0 or facePos is None:
306                     dontlook += 1
307                     if dontlook % 20 is 0:
308                         print('Conducteur inattentif')
309                         sendSerial(ser,'s')
310                 else:
311                     dontlook = 0
312                     if i < 10:
313                         i += 1
314                     if i is 10:
315                         self.initNeutral(frame)
316                         i += 1
317                     frame = self.drawDetected(frame, facePos, (0,140,255))
318                     cropped = self.cropFromFace(frame, facePos)
319                     eyePos = self.getCroppedEyesPos(cropped)
320                     sendSerial(ser,'d')
321                     if eyePos is None:
322                         #print('yeux fermes ou yeux non detectes')
323                         sleep += 1
324                         error = 1
325                     elif error is not 1:
326                         sleep = 0
327                         sendSerial(ser,'d')
328                         sendSerial(ser,'r')
329                     else:
330                         error = 0
331                     if sleep > 3:
332                         print('Endormi')
333                         sendSerial(ser,'w')
334                         sendSerial(ser,'s')
335                     if eyePos is not None and i > 10:
336                         cropped = self.drawDetected(cropped, eyePos, (255,0,255))
337                         eyeFrame = self.cropFromFace(frame, eyePos)
338                         if self.isEyeClosed(eyeFrame):
339                             #print('yeux fermes ou yeux non detectes')
340                             sleep += 1
341                             error = 1
342                         elif error is not 1:
343                             sleep = 0
344                         else:
345                             error = 0
346                         if len(eyePos) > 0 and self.EyeHeightThanNeutral(eyePos[0][3], 5):
347                             #print('plus grand')
348                             eyeBigger += 1
349                             error = 1
350                         elif error is not 1:
```

```
351                         eyeBigger = 0
352                     else:
353                         error = 0
354                     if len(eyePos) > 0 and self.EyeNotHeightThanNeutral(eyePos[0][3], 4):
355                         #print('plus petit')
356                         eyeNotBigger += 1
357                         error = 1
358                     elif error is not 1:
359                         eyeNotBigger = 0
360                     else:
361                         error = 0
362                 mouthPos = self.getCroppedMouthPos(cropped)
363                 cropped = self.drawDetected(cropped, mouthPos, (0,0,255))
364                 if mouthPos is not None and self.isMouthOpen(self.cropFromFace(frame, mouthPos)) and
365                     #print('bouche ouverte')
366                     mouthOpen += 1
367                     error = 1
368                 elif not self.isMouthOpen(self.cropFromFace(frame, mouthPos)) and error is not 1:
369                     moutOpen = 0
370                 else:
371                     error = 0
372 
373                 if mouthOpen > 5 and eyeBigger > 5:
374                     print('Surpris')
375                     sendSerial(ser,'b')
376                 else:
377                     sendSerial(ser,'n')
378                 if eyeNotBigger > 5:
379                     print('Enerve')
380                     sendSerial(ser,'a')
381                 else:
382                     sendSerial(ser,'q')
383                 #if mouthOpen <= 5 and eyeNotBigger > 5:
384                 #    print('enerve')
385                 #    sendSerial(ser,'a')
386                 cv2.imshow("Tete", cropped)
387             cv2.imshow("TIPEAPS", frame)
388             key = cv2.waitKey(20)
389             if key in [27, ord('Q'), ord('q')]:
390                 break
391 
392     def capture(self):
393         """Recupere le flux video"""
394         interval = 0
395         if self.camera.isOpened():
396             (rval, frame) = self.camera.read()
397         else:
398             rval = False
399         i = 0
400         while i < 55:
401             i+=1
402             (rval, frame) = self.camera.read()
403             self.time = time.time()
404             facePos = self.getFacesPos(frame)
405             frame = self.drawDetected(frame, facePos, (0,140,255))
406             for f in facePos:
407                 x,y,w,h = [v for v in f]
408                 resized = self.extractAndResize(frame, x, y, w, h)
409                 identity, confidence = self.identify(resized)
410                 if confidence > THRESHOLD:
411                     identity = "INCONNU"
412                 INDIVIDUS.append(identity)
```

```
413                 print(identity + "␣---␣" + str(confidence))
414                 cv2.putText(frame, "%s␣(%s)"%(identity, confidence), (x,y), cv2.FONT_HERSHEY_PLAIN,
415             cv2.imshow("TIPEAPS", frame)
416             key = cv2.waitKey(20)
417             if key in [27, ord('Q'), ord('q')]: #esc / Q
418                 break
419
420 def getSerialName():
421     """Retourne le fichier correspondant a l'arduino"""
422     serialName = '/dev/null'
423     osname = sys.platform.lower()
424     if 'darwin' in osname: #si mac OS X
425         for tty in glob.glob('/dev/tty*'):
426             if 'usbmodem' in tty:
427                 serialName = tty
428     elif 'linux' in osname: #si linux
429         for tty in glob.glob('/dev/tty*'):
430             if 'ACM' in tty:
431                 serialName = tty
432     return serialName
433
434 if __name__ == "__main__":
435     mode = 0
436     for i in range(1,len(sys.argv)):
437         if sys.argv[i] == '-n' and i < len(sys.argv):
438             individu = sys.argv[i + 1]
439         if sys.argv[i] == '-a':
440             ARDUINO = True
441             ser = serial.Serial(getSerialName(), 9600)
442         if sys.argv[i] == '-c':
443             mode = 1
444     if mode is 0:
445         recognize = Recognize("images")
446         recognize.recognize()
447         d = defaultdict(int)
448         for i in INDIVIDUS:
449             d[i] += 1
450         result = max(d.iteritems(), key=lambda x: x[1])
451         individu = result[0]
452         if result[1] > 25:
453             print('Bienvenue␣' + individu)
454             if ARDUINO:
455                 sendSerial(ser, 'e')
456             recognize.emotions()
457     if mode is 1:
458         createDB = CreateDataBase("images", individu)
459         createDB.capture()
```

## A.2 Code Arduino pour les tests

```
1  /**
2   * oooooooooo   ooooo   oooooooooo ooooooooooo      o       oooooooooo   oooooooo8
3   *  88 888  88    888    888     888 888              888       888    888 888
4   *     888         888    888oooo88  888oooo8        8  88      888oooo88   888ooooooo
5   *     888         888    888         888           8oooo88     888                888
6   *    o888o       o888o  o888o       o888ooo8888 o88o  o8888o o888o       o88oooo888
7   */
8
9  int pinEngine = 7;
10 int pinWarning = 6;
11 int pinAccelerationLimit = 5;
```

```
12  int pinBrake = 4;
13  int pinSound = 8;
14
15  boolean engine = false;
16
17  char command = 0;
18
19  void Exit(){
20    engine = false;
21    digitalWrite(pinEngine,LOW);
22    digitalWrite(pinWarning,LOW);
23    digitalWrite(pinAccelerationLimit,LOW);
24    digitalWrite(pinBrake,LOW);
25
26    noTone(pinSound);
27  }
28
29  void Engine(){
30    engine = true;
31    digitalWrite(pinEngine, HIGH);
32  }
33
34  void Warning(){
35    digitalWrite(pinWarning, HIGH);
36  }
37
38  void StopWarning(){
39    digitalWrite(pinWarning, LOW);
40  }
41
42  void AccelerationLimit(){
43    digitalWrite(pinAccelerationLimit, HIGH);
44  }
45
46  void StopAccelerationLimit(){
47    digitalWrite(pinAccelerationLimit, LOW);
48  }
49
50  void Brake(){
51    digitalWrite(pinBrake, HIGH);
52  }
53
54  void StopBrake(){
55    digitalWrite(pinBrake, LOW);
56  }
57
58  void Sound(){
59    tone(pinSound, 666);
60  }
61
62  void StopSound(){
63    noTone(pinSound);
64  }
65
66  void setup()
67  {
68    pinMode(pinEngine, OUTPUT);
69    pinMode(pinWarning,OUTPUT);
70    pinMode(pinAccelerationLimit,OUTPUT);
71    pinMode(pinBrake,OUTPUT);
72
73    Serial.begin(9600); //On demarre la connexion serie
```

```
74    while(!Serial){} // on attend que la connexion serie demarre
75
76    /**
77     *  verification du fonctionnement des systemes
78     *  (1/2 seconde)
79     */
80    digitalWrite(pinEngine, HIGH);
81    Warning();
82    AccelerationLimit();
83    Brake();
84    Sound();
85
86    delay(500);
87
88    digitalWrite(pinEngine, LOW);
89    StopWarning();
90    StopAccelerationLimit();
91    StopBrake();
92    StopSound();
93  }
94
95  void loop()
96  {
97    if(!engine)//si non demarre
98    {
99      command = Serial.read();
100     if(command == 'e')
101       Engine();
102     command = 0;
103   }
104   else
105   {
106     if(Serial.available() > 0) //si on recoit une donnee sur le port serie
107     {
108       command = Serial.read();
109       switch(command){
110         case 'x':
111           Exit();
112         break;
113         case 'w':
114           Warning();
115         break;
116         case 'r':
117           StopWarning();
118         break;
119         case 'b':
120           Brake();
121         break;
122         case 'n':
123           StopBrake();
124         break;
125         case 'a':
126           AccelerationLimit();
127         break;
128         case 'q':
129           StopAccelerationLimit();
130         break;
131         case 's':
132           Sound();
133         break;
134         case 'd':
135           StopSound();
```

```
136             break;
137           }
138           command = 0;
139         }
140       }
141   }
```