Sommaire

Liste Des Figures

1. Cadre du projet :

Les bandes de guidage au sol et les feux sonorisés sont de plus en plus répandus pour aider les personnes aveugles ou malvoyantes à se repérer dans les espaces publics comme les zones piétonnes, les routes, les gares, les arrêts de bus.

Ainsi dans ce cadre on a choisit notre projet intitulé : **Système d'aide pour les personnes malvoyantes et aveugles.**

2. Introduction :

L'objectif du projet est de proposer une caméra intelligente qui permet de traduire une image qui contient un texte en un son clair et audible, d'autre fonctions peuvent d'être ajouter en faisant la traduction de langage de prononciation telque convertir un texte écrit en Espagnol en un son adapté au Français.

Ce projet est basé sur une carte Raspberry Pi. Elle contrôle plusieurs périphériques tels que entrés sortie , caméra, haut-parleur et LCD qui agissent comme une interface entre le système et l'utilisateur.

Son principal objectif est de faire la reconnaissance optique de caractères qui sera implémentée pour reconnaître les caractères qui sont ensuite lus par le système via un haut-parleur.

Notre projet a comme avantage son faible coût (carte Raspberry + camera + haut de parleur), d'autre part il pourra être utilisé dans plusieurs cas :

1. La lecture des affiches des arrêts de bus pour les personnes non voyantes..
2. Traduction instantané des documents.
3. La lecture des livres avec une bonne monotonie pour les personnes qui ont des problèmes.
4. Aides des étudiants malvoyant pour les études ...

3. Outils de développement :

3.1 Outils Software:

Pour la réalisation de notre projet on a utilisé les outils logiciels suivants :

Python3 : C'est un langage de programmation objet, multi paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions.

Figure 1 : Symbole Python 3

Pour pouvoir faire le processing de l'image envoyé par le camera et la transformer en texte on a installé **OpenCV**et pytesseract.

OpenCV : C' est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel. La bibliothèque OpenCV met à disposition de nombreuses fonctionnalités très diversifiées permettant de créer des programmes partant des données brutes pour aller jusqu'à la création d'interfaces graphiques basiques.

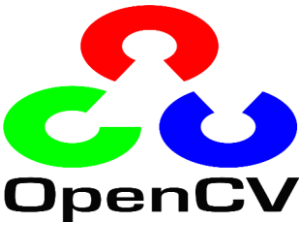

Figure 2 : Symbole OpenCV

Pytesseract : Python-tesseract est un wrapper pour Tesseract-OCR de google.

Il est également utile en tant que script d'appel autonome à tesseract de lire tous les types d'image pris en charge par la bibliothèque d'imagerie Python, y compris jpeg, png, gif, bmp, tiff, et d'autres, où tesseract-ocr par défaut ne supporte que tiff et bmp.

De plus, s'il est utilisé en tant que script, Python-tesseract affichera le texte au lieu de l'écrire dans un fichier.

3.2 Outils Hardware:

3.2.1 Carte Raspberry pi 3 :

Lancé en 2013, le Raspberry Pi est un nano ordinateur détaillé à environ . La dernière version, le Raspberry Pi 3, possède un processeur ARM de 1200 MHz, 4 ports USB, 1 port Ethernet, un module Bluetooth LE, un port HDMI, une prise audio 3.5 mm et un lecteur de carte Micro SD.

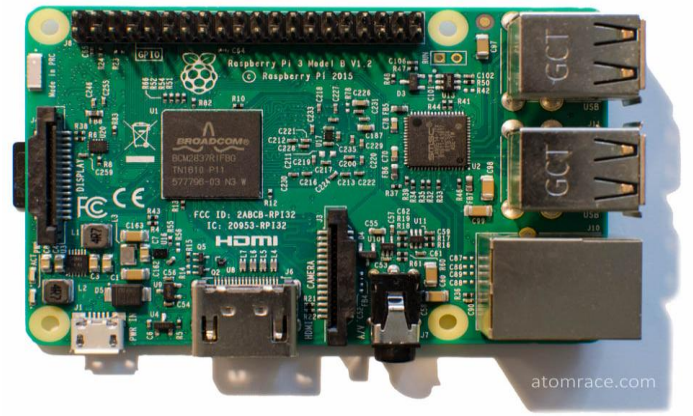

Figure 3 : Carte Raspberry pi 3

Le Raspberry Pi offre aussi des ports GPIO permettant au technophile d'y brancher différents modules matériels. Il est donc possible d'y connecter des capteurs (sensors) transformant ce petit ordinateur en plate-forme idéale pour l'Internet des Objets.

2.1.2 Caméra compatible Raspberry :

- Haute résolution et images claires grâce à une caméra de 5 mégapixels.
- Grâce aux filtres infrarouges, cette caméra vidéo et caméra photo peut faire des images superbes également en lumière vive.
- Câblage facile de la caméra avec un Raspberry Pi grâce à un câble plat.

Figure 4 : Caméra Raspberry 5 méga Pixel

Pour recevoir le son on a eu besoin aussi d'un Haut de parleur pour écouter le son émis par le port de sortie audio de la carte.

4. Démarche de conception

4.1 Configuration pour l'accès à la carte Raspberry par cable Ethernet

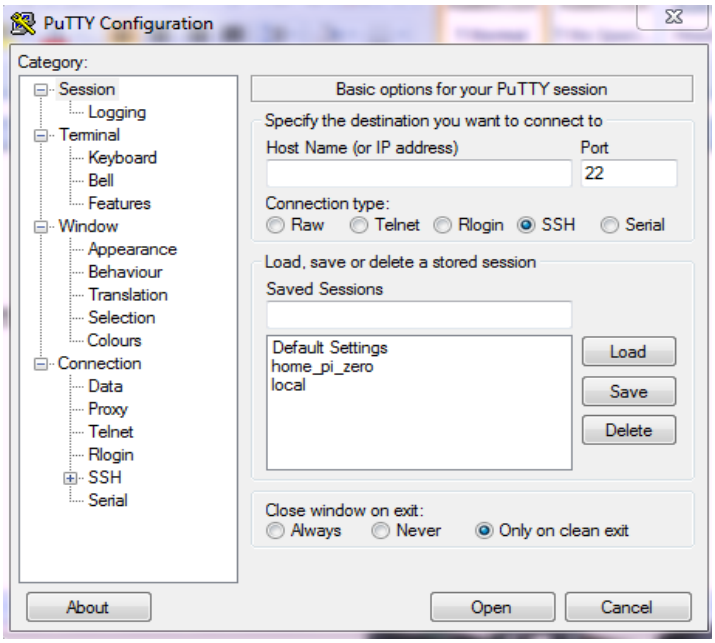

Figure 5 : Outil ssh pour Raspberry Pi

pour rendre l'accès à la carte Raspberry facile il faut soit utiliser l'outil de communication par ssh Putty (Figure 5) soit l'outil bureau à distance (Remote Desktop) sous windows (Figure 6).

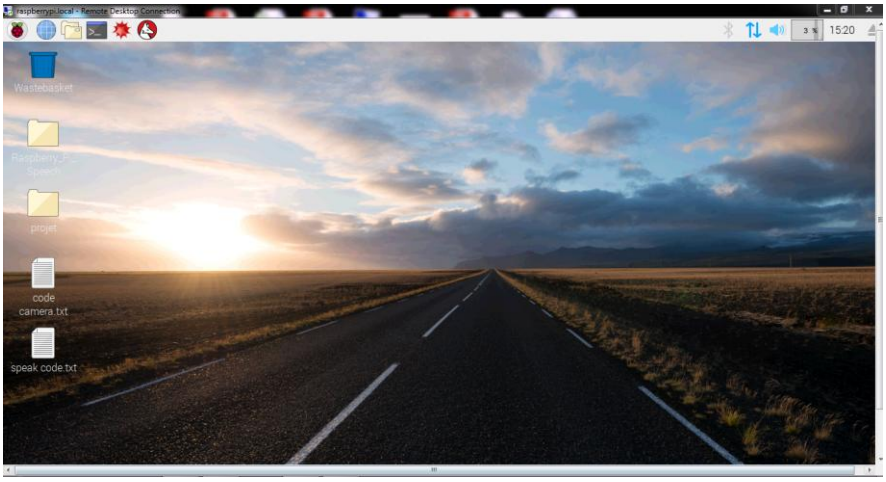

Figure 6 : Outil bureau à distance pour raspberry Pi sous Windows

4.2 Installation de la caméra

La caméra est compatible avec la carte de développement Raspberry , elle est de résolution 5 méga pixel, son interfaçage est facile avec la carte.

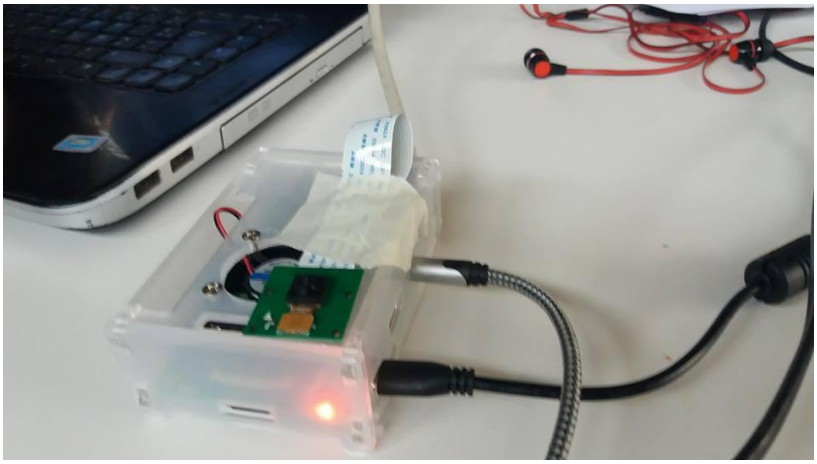

Figure 7 : Interfaçage caméra carte Raspberry 3 et haut parleur

Pour activer la caméra il faut taper la commande **sudo raspi-config,** ensuite choisir **Interfacing Options**, et cocher la section **camera enable.**

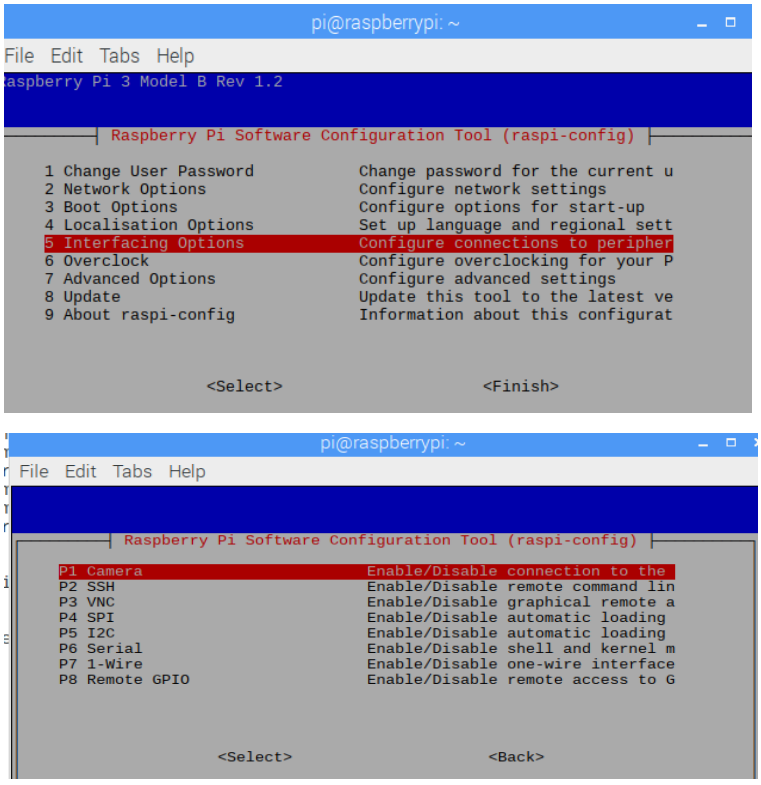

Figure 8 : Interfaçage et activation de la camera

4.3 Installation de la bibliothèque Open Cv et python :

```
sudo apt-get install python3 python3-setuptools python3-dev -y
wget https://bootstrap.pypa.io/get-pip.py
sudo python3 get-pip.py
sudo pip3 install numpy
```

Figure 9: Installation de l'outil de programmation Python 3

```
cd ~/opencv-3.3.0
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D INSTALL_C_EXAMPLES=OFF \
    -D INSTALL_PYTHON_EXAMPLES=ON \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.3.0/modules \
    -D BUILD_EXAMPLES=ON ..
```

Figure 10 : Installation des modules supplémentaires OpenCV 3.4 et contrib :

```
cd opencv-3.4.0
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D BUILD_opencv_java=OFF \
-D BUILD_opencv_python2=OFF \
-D BUILD_opencv_python3=ON \
-D PYTHON_DEFAULT_EXECUTABLE=$(which python3) \
-D INSTALL_C_EXAMPLES=OFF \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D BUILD_EXAMPLES=ON\
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.4.0/modules \
-D WITH_CUDA=OFF \
-D BUILD_TESTS=OFF \
-D BUILD_PERF_TESTS= OFF ..
```

Figure 11 : **Compilation et installation de la bibliothèque OpenCV 3.4.0 pour Python 3 :**

```
pi@raspberrypi:~ $ python3
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.4.0'
```

Figure 12 : Tester le bon déroulement de l'installation d'OpenCV 3.4.0

4.4 Capture d'une image à travers la caméra

Après l'installation de la bibliothèque « Pi camera » Le code suivant permet la capture d'une image avec une résolution **1024x720** :

```
with picamera.PiCamera() as camera:
 camera.resolution = (1024, 720)
 camera.capture("f1.jpg")
 print("picture taken.")
```

Figure 13 : Programme pour capture d'une image à l'aide de la caméra

4.5 Extraction du texte depuis l'image capturée par la caméra.

```
def get_string(img_path):
    Read image with opencv

    capture = cv2.imread(img_path)
    Convert to gray
    capture = cv2.cvtColor(capture, cv2.COLOR_BGR2GRAY)

    Apply dilation and erosion to remove some noise
    kernel = np.ones((1, 1), np.uint8)
    img = cv2.dilate(img, kernel, iterations=1)
    img = cv2.erode(img, kernel, iterations=1)

    Write image after removed noise
    cv2.imwrite("removed_noise.jpg", img)

     Apply threshold to get image with only black and white
    img = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 31, 2)

    Write the image after apply opencv to do some ...
    cv2.imwrite("capture.jpg", capture)

    Recognize text with tesseract for python



    result = pytesseract.image_to_string(Image.open("img.jpg"))

    Remove template file
    os.remove(temp)

    return result

time.sleep(5)


print ('--- Start recognize text from image ---')
```

Figure 14 : Extraction de texte depuis l'image r

La variable résultat contient le texte déduit à partir de l'image capturée.

4.6 Conversion du texte et génération du fichier audio.

Pour convertir un texte il faut installer les deux bibliothèque s suivantes : gtts

(Google text to speech) et la mpg321 pour la génération d'un fichier audio de type mp3. On peut fixer la langue pour la prononciation en anglais.

```
print ('--- Start recognize text from image ---')

result1=get_string("img.jpg")
print (result1)
time.sleep(5)

tts = gTTS(text=result1 , lang='en')
tts.save("result.mp3")
os.system("mpg321 result.mp3")


print ("------ Done -------")
```

Figure 15 : Conversion du texte en fichier audio mp3

4.7 Compilation et résultat

```
pi@raspberrypi:~/Desktop/Project $ python3 recognition2.py
--- Start recognize text from image ---
picture taken.
3 years later and I'm actually doing great. I have my own
plaoe, a decent job, my pets are doing very well My credit
is shit from the divorce but really, I'm in decent company in
that. This isn't really a sob story; I'm really proud of what I've
done on my own. | Just wanted to share for possibly some
closure since I never really got that. Strangers on the
internet are better than keeping it inside for so long. And if
you hate my story ljust lose some fake points anyway. I
save these text messages to remind myself how far I've
come, not to cry over. I'm still terrible in relationships but. .
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layer 1, 2, and 3.
Version 0.3.2-1 (2012/03/25). Written and copyrights by Joe Drew,
now maintained by Nanakos Chrysostomos and others.
Uses code from various people. See 'README' for more!
THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTY! USE AT YOUR OWN RISK!

Playing MPEG stream from result.mp3 ...
MPEG 2.0 layer III, 32 kbit/s, 24000 Hz mono

[0:47] Decoding of result.mp3 finished.
------ Done -------
```

Figure 16 : Compilation du programme et affichage du résultat.

Nous avons mis dans le code un temps d'attente de 5 s avant la prise de l'image afin d'avoir la meilleur position pour la capture.

Pour s'assurer de la bonne extraction du texte depuis une image on affiche le texte sur le terminale avant la démarrage de lecture.

Le fichier audio est sauvegarde au format mp3 puis on a mis le code permettant sa lecture automatique.

5. Conclusion

Après la réalisation d'un prototype fonctionnelle, nous avons trouvé qu'il est indispensable d'ajouter un bouton extérieur et une batterie pour que le système soit autonome et portable.

Ce prototype est utilisé pour tester la faisabilité du projet puis nous avons cherché à ajouter une deuxième partie au niveau du code pour que le système sera autonome**.**