Implémentation de données sur une interface caméra en utilisant Raspberry Pi et une caméra embarquée

Salim Khazem

20 octobre 2020

Introduction

L'amélioration et le développement des dispositifs électroniques et les systèmes embarqués ont engendré de nouveaux besoins d'interaction des dispositifs embarqué et l'environnement extérieur, En effet l'apparition des interfaces hommes machines a permis une amélioration conséquente de la diffusion d'informations de capteurs.

Dans ce document, nous allons nous intéresser à la diffusion de données sur une interface d'une caméra en temps réel, pour cela nous allons utilisé des composants embarqués, notamment un capteur pour illustrer cette implémentation.

Composants utilisés

— **Raspberry Pi 3**

Pour faire la démonstration, nous avons utilisé une carte Raspberry pi modèle B3, le schéma de la carte est ci-dessous :

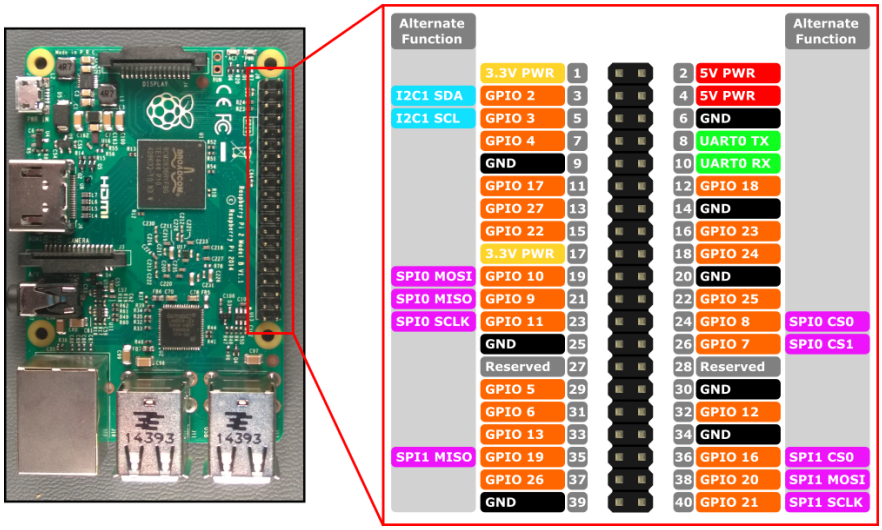

FIGURE 1 – Raspberry Pi 3

— Caractéristique de la carte :

- Processeur Cortex-A53 64-bit à 1.4GHz
- 1 Gb de SDRAM
- 40 GPIO Headers
- Interface vidéo (HDMI, Display, Caméra)
- Emplacement Carte SD

— **Pi Camera**

Pour l'implémentation de notre interface, nous avons choisit la Pi Camera, qui est une caméra adaptée à la Raspberry Pi :

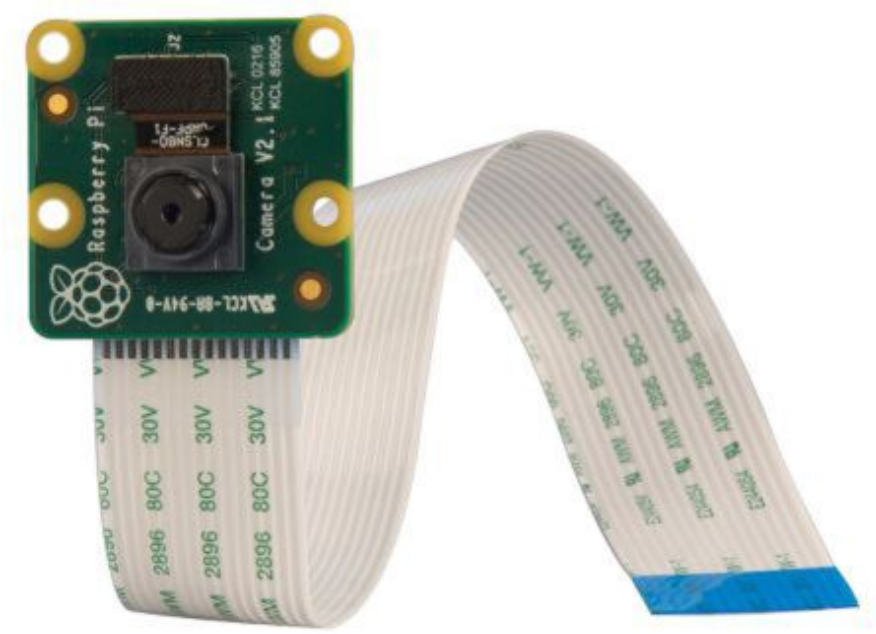

FIGURE 2 – Pi Camera

— Caractéristique de la PiCamera :

- Résolution 8 Mégapixels
- Format de la vidéo (1080p, 720p, 480p)
- Résolution du capteur (2592 × 1944 pixels)

— **Capteur Ultrason HC-SR04**

Pour le capteur, le choix s'est porté sur celui ci, car je l'avais sous la main, et son implémentation est assez simple, et le nombre d'applications qu'on peut avec sont diverses . Le SR04 est un capteur numérique, qui possède 4 pins, ce dernier utilise des ultrasons pour déterminer la distance d'un objet (*HC-SR04 Datasheet*, n.d.).

— Caractéristique du capteur HC-SR04 :

- Plage de mesure : 2 cm à 400 cm
- Résolution de la mesure 0.3 cm
- Angle de mesure efficace : 15°

- Largeur d'impulsion sur l'entré du déclenchement : 10 µs

FIGURE 3 – Capteur Ultrason HC-SR04

Installation de l'environnement

Pour cette partie, le choix s'est porté sur une carte Raspberry pi 3, cela est dû à sa performance de calcul qui nous permet de faire du traitement d'image dessus, le choix est également dû à la richesse de librairies qu'on trouve sur ces cartes là mais également son environnement. En effet, c'est une carte qui embarque du Linux (distribution Debian) comme OS, la carte peut être programmée avec le langage Python, qui est un atout pour le traitement d'images.

Pour ce faire, nous allons en premier lieu, installer les packages nécessaires :

Pour commencer il faudra installer le package OpenCv. qui est une bibliothèque très riche développée par Intel pour le traitement d'images et cela se passe comme suit :

1. Ouvrir un terminal sur la Raspberry
2. Lancer la commande : sudo pip install opencv-python (dans le cas ou vous êtes sur Python 2.7)
3. Pour la version 3.x il faudra lancer la commande : sudo pip3 install opencv-python
4. Après avoir installer le package, pour vérifier que cela s'est fait correctement, on lance une fenêtre python sur le terminal et on lance la commande : import cv2 .

PS : à l'installation du package OpenCV, d'autres packages seront également installés

Initialisation de la PiCamera

Pour initialiser la caméra, nous allons tester un petit script qui allumera la PiCamera pour voir si cette dernière est bien fonctionnelle.

Le script ci-dessous permet de tester à la fois la caméra et de prendre une photo afin de la stocker dans le répertoire Desktop

```
# Permet d'importer la bibliothèque nécessaire pour l'utilisation de la PiCamera

from picamera import PiCamera


#Importation de la fonction sleep(pour le délais ) de la bibliothèque time

from time import sleep


#Instanciation de l'objet PiCamera que nous allons utilisé
camera = PiCamera()

camera.start_preview()

sleep(5)

camera.capture('/home/pi/Desktop/TestCamera.jpg')

camera.stop_preview()
```

Implémentation de données dans l'interface de la PiCamera

Pour interfacer les pin I/O de la Raspberry Pi, il faut utiliser la bibliothèque RPi.GPIO,

— **Acquisition de la donnée**

Pour ce faire, il faut en premier lieu initialiser les pin du capteur comme suit :

— Trigger : en Output
— Echo : en Input

Sur la figure ci-dessous, le principe de fonctionnement est explicité.

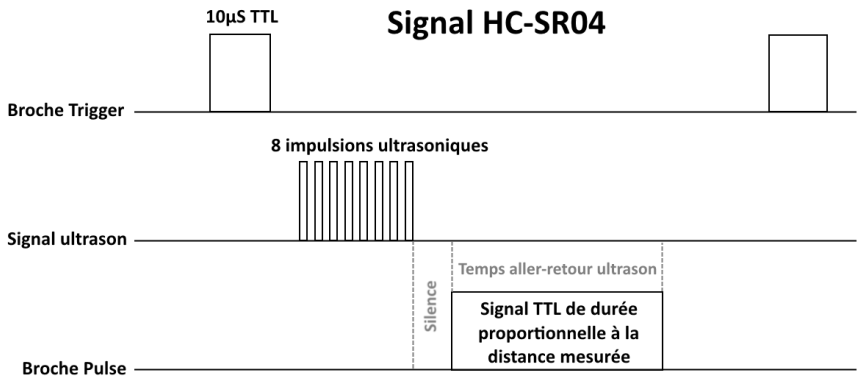

FIGURE 4 – Fonctionnement du capteur Ultrason

— **Le principe de fonctionnement**

Le principe de fonctionnement de ce capteur est entièrement basé sur la vitesse du son, et cela se déroule comme suit :

1. On envoie une impulsion HIGH de 10µs sur la pin Trigger du capteur (permet l'actionnement du capteur) .
2. Le capteur envoie une série de 8 impulsions ultrasoniques à 40KHz .
3. Les ultrasons se propagent dans l'air jusqu'à toucher un obstacle et retourne dans l'autre sens vers le capteur.
4. Le capteur détecte l'Echo et clôture la mesure

La pin Echo reste en HIGH pendant l'étape 3 et 4 , ce qui nous permet de mesurer le temps d'un aller-retour des ultrasons et donc déterminer la distance.

— **Interfaçage de la donnée sur la PiCamera**

Après l'acquisition de donnée du capteur, il nous reste à les affiché sur l'interface de la PiCamera,

pour ce faire on utilise la bibliothèque PiCamera ainsi que OpenCV , on fait un script ci-dssous :

```
"""
Ceci est un commentaire
Sur la première partie on importe les bibliothèque nécessaire pour notre application
* cv2 : nous permet d'afficher et de faire du traitement sur la vidéo
* datetime : nous permet d'afficher la date et l'heure actuele
* PiCamera : gère la Picamera ainsi que ses drivers
* RPi.GPIO : nous permet d'intéragir avec l'environnement extérieur I/O
* time : nous permet d'utiliser les fonctions de l'horloge (délais, interruption, calcul d'intervale de
"""
import cv2
import datetime as dt
from picamera import PiCamera
import time
import RPi.GPIO as GPIO
```

```
import os


# définition du mode d'affectation des pin (BCM) utilise les annotation GPIO
GPIO.setmode(GPIO.BCM)
TRIG=22  #Affectation de la pin Trigger du capteur à la pin GPIO 22
ECHO =13  #Affectation de la pin Echo du capteur à la pin GPIO 13

GPIO.setup(TRIG,GPIO.OUT) #Affectation du TRIG en OUTPUT
GPIO.setup(ECHO,GPIO.IN)  #Affectation de ECHO en INPUT

#Initialisation de la caméra l'argument 0 est le périphérique utilisé
cam = cv2.VideoCapture(0)
while True:

    ret,frame = cam.read() # Acquisition de la données vidéo de la caméra

    #Envoie d'impulsion pendant 10 \selectlanguage{greek}µ\selectlanguage{english}s

    GPIO.output(TRIG,GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(TRIG,GPIO.LOW)

    #Attente de l'echo
    while GPIO.input(ECHO) == 0:
        pass
    start=time.time() #Initialisation du temps juste apr\selectlanguage{ngerman}ès l'envoie des impulsi

    while GPIO.input(ECHO) == 1:
        pass

    stop=time.time() # Calcul de temps juste après reception de l'echo (Compteur)
    distance = (stop-start) * 17000  # Opération qui nous permet de convertir le temps en distance

    dist="Distance :  " + str(distance)  #Conversion de la donnée distance en String

    font = cv2.FONT_HERSHEY_SIMPLEX  #Format du texte

    #Fonction nous permettant de mettre du texte sur l'interface caméra

    cv2.putText(frame,
                dist,
                (50, 50),
                font, 1,
                (0, 0, 255),
                2,
                cv2.LINE_4)
    cv2.putText(frame,str(dt.datetime.now()),(50,20),font,1,(255,255,255),
        2,cv2.LINE_4)
    cv2.imshow('PiCam',frame)  # fonction permettant d'afficher une fenêtre de la caméra
```

```
    #condition pour pouvoir interrompre et quitter la boucle

    if cv2.waitKey(1) == ord('q') :
        break

#Arrêt de l'acquisition de la donnée image et fermeture de la fenêtre
cam.release()
cv2.destroyAllWindows()
```

En lançant le programme, on aura une interface équivalente à celle se trouvant ci-dessous

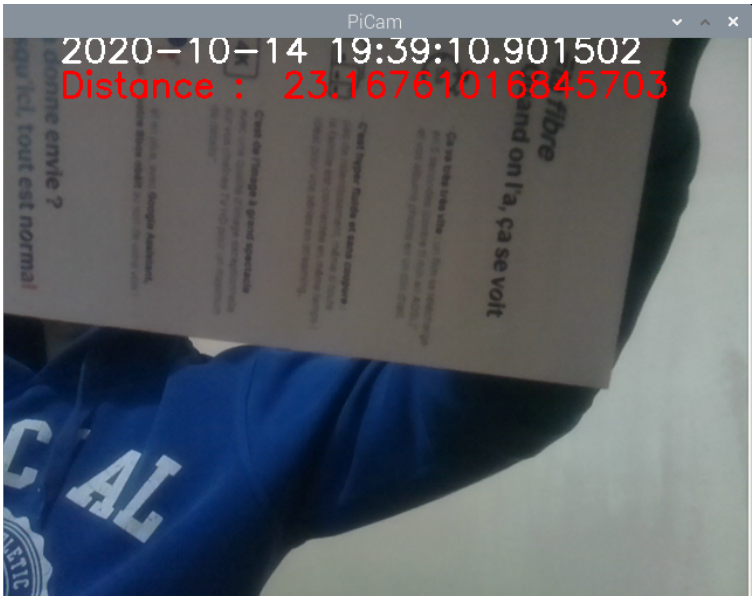

FIGURE 5 – Simulation de l'interface

Toutefois, la PiCamera et OpenCv permettent également de sauvegarder la vidéo sur la mémoire de la Raspberry et bien plus encore.

Concernant l'encodage de la vidéo, on peut principalement utiliser deux sur ce genre d'application, le H264 et le Mjpeg

et pour l'image, on peut utiliser (jpeg,png, bgr, gif . . .) et bien plus encore.

Diagramme du traitement de la vidéo

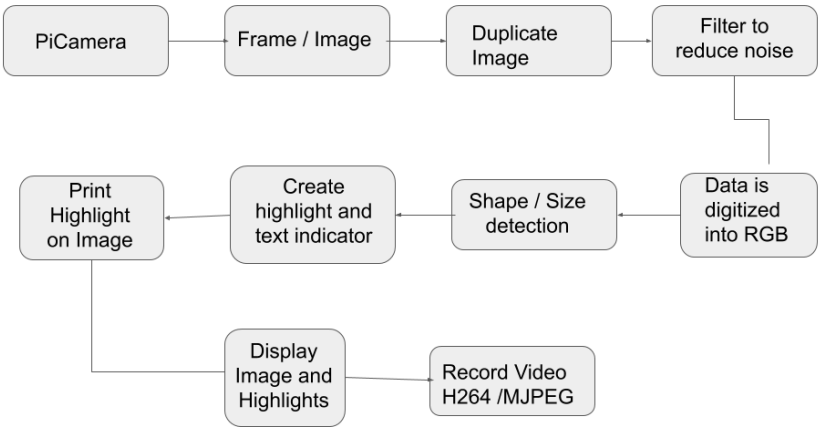

FIGURE 6 – Diagramme du traitement de la vidéo

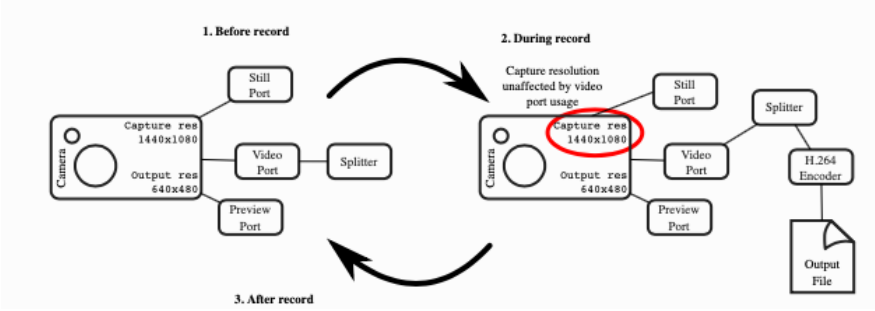

FIGURE 7 – Diagramme simplifié du traitement de la vidéo

Références

http://web.eece.maine.edu/~zhu/book/lab/HC-SR04%20User%20Manual.pdf. http://web.eece.maine.edu/~zhu/book/lab/HC-SR04%20User%20Manual.pdf