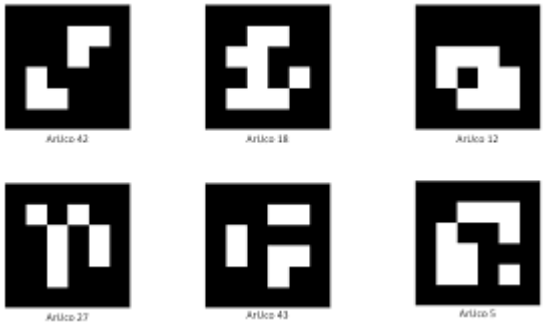

Artico 42
Artico 18
Artico 12
Artico 27
Artico 43
Artico 5

ARUCOCODE

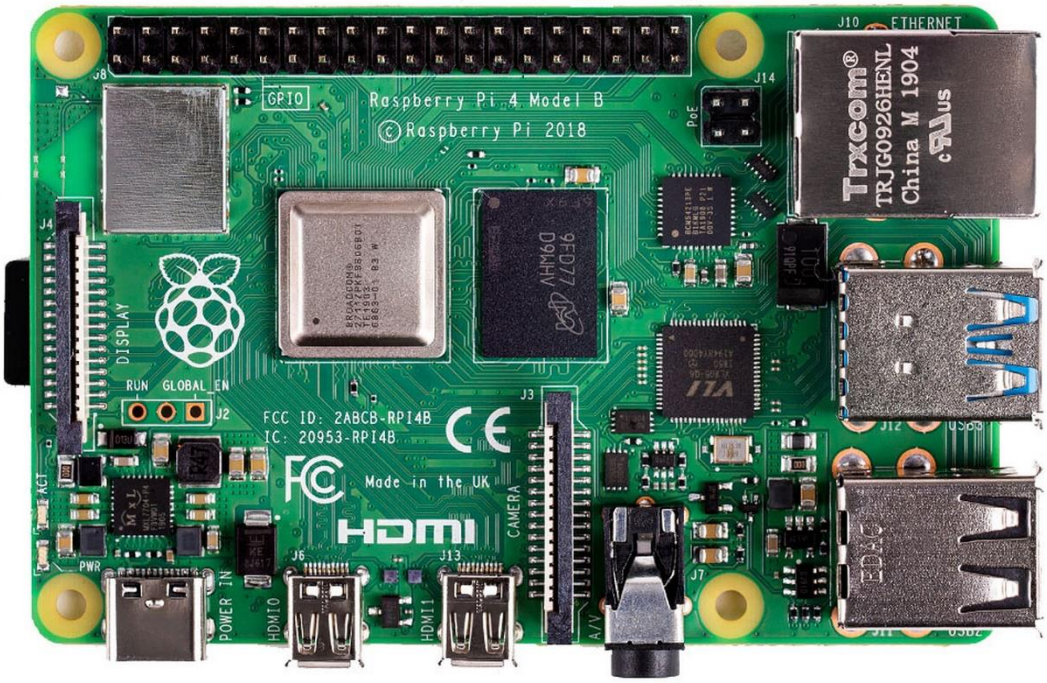

Raspberry Pi 4 Model B
©Raspberry Pi 2018
GPIO
ETHERNET
Trxcom®
TRJG0926HENL
China M 1904
DISPLAY
RUN
GLOBAL_EN
FCC ID: 2ABCB-RPI4B
IC: 20953-RPI4B
Made in the UK
HDMI
CAMERA
POWER IN
HDMI0
HDMI1
PWR
A/V

Table des matières

Matériels :

-Raspberry Pi 4

-Ecran avec son cable

-Carte micro-SD

-Adaptateur Carte microSD

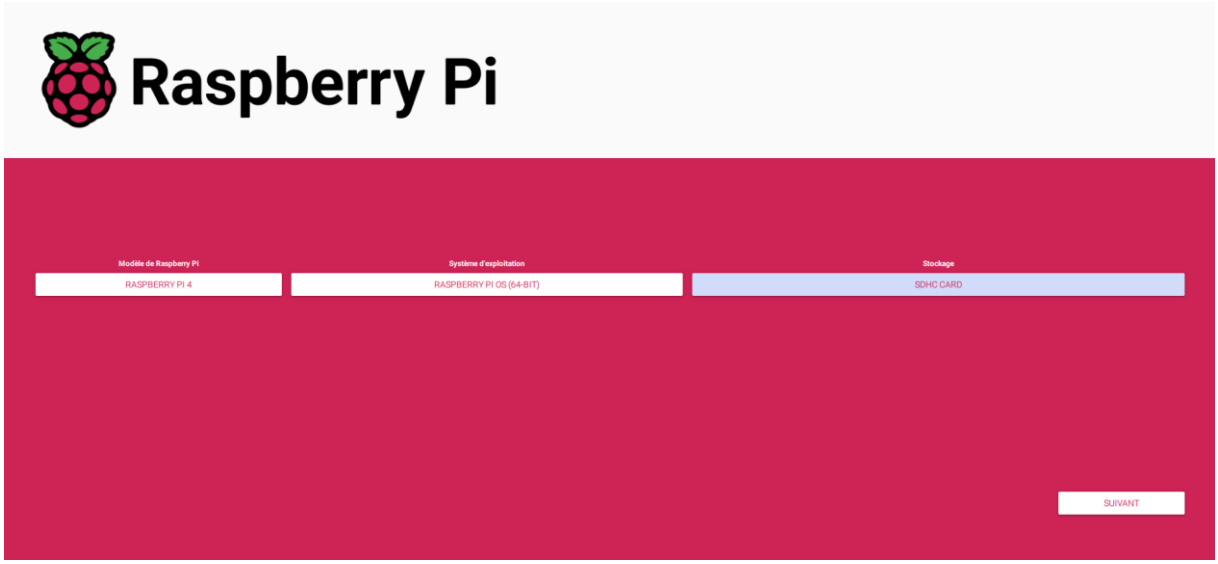
-Un clavier

-Une souris

Installation de l'OS Raspberry :

Sur ton ordinateur, tu dois installer l'application Raspberry Pi Imager.

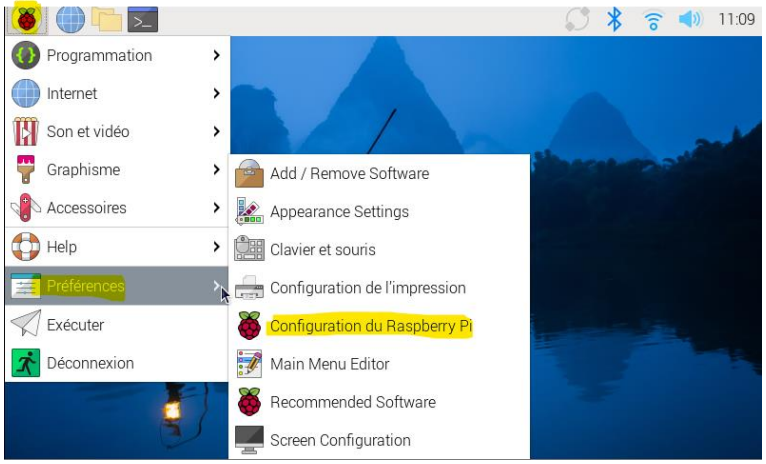
- Sélectionne le model de ton Raspberry.
- Sélectionne l'OS du Raspberry
- Sélectionne la carte



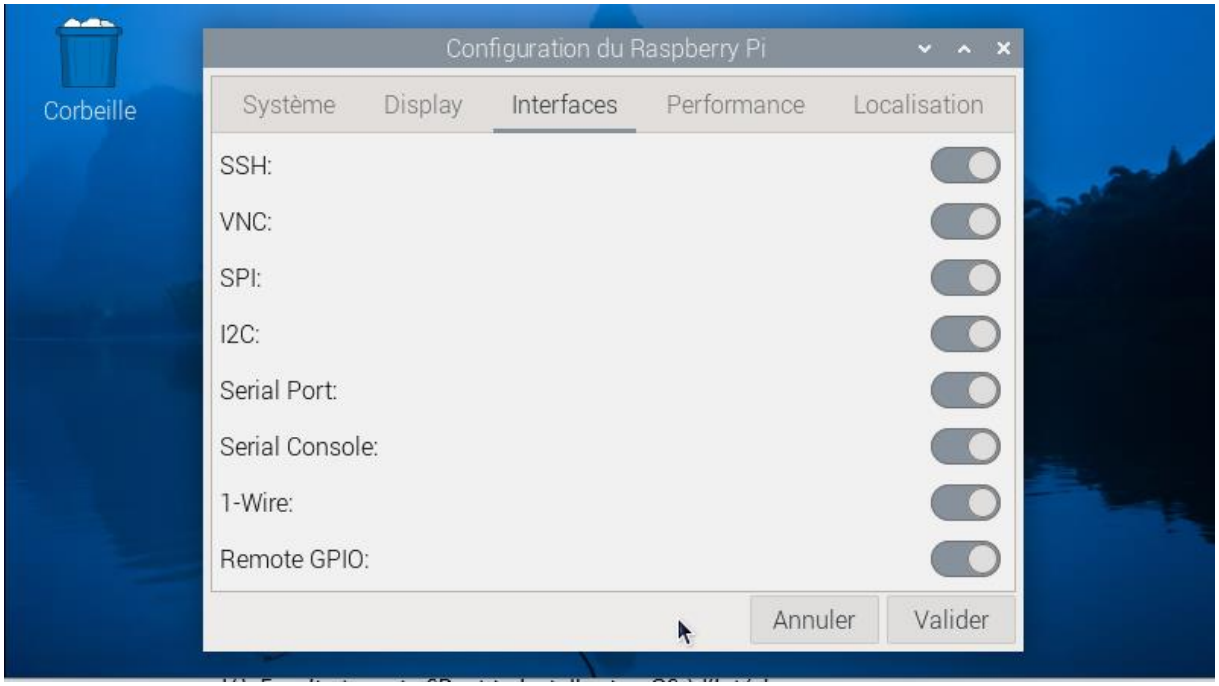
1er démarrage du Raspberry :

Lors du 1er démarrage, tu configures la langue, tes identifiants, la connexion et le navigateur par défaut.
Ensuite, tu fais la mise à jour.

Puis tu vas dans l'icone Raspberry en haut à droite < Préférences < Configuration du Raspberry Pi.

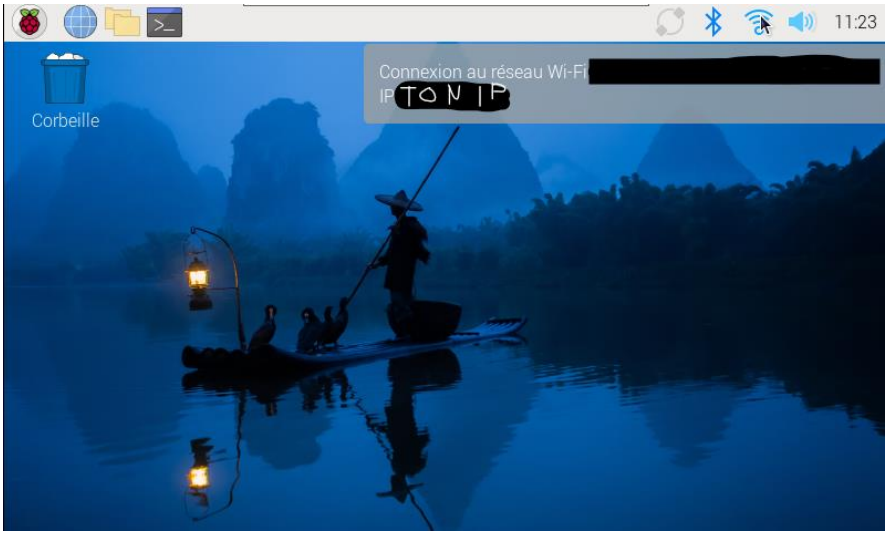


Tu vas dans interfaces, tu mets tout en ON et redémarre.



Conseil :

Si tu veux, tu peux installer Real VNC Viewer pour contrôler ton Raspberry depuis ton ordinateur pour faciliter les copier-coller. Ton ordinateur doit être sur le même réseau que ton Raspberry. Mets ta souris sur l'icone WIFI pour connaitre l'IP de ton Raspberry.



Sur ton ordinateur, sur RealVNC Viewer, tu mets ton IP puis appuie sur la touche entrée. Cela va créer une fenêtre tu devras cliquer dessus et mettre tes identifiants. Puis pour être connecté au Raspberry, tu cliqueras plusieurs fois sur la fenêtre jusqu'à que cela s'ouvre.

Test de la caméra :

Pour voir si la caméra fonctionne, tu ouvre le terminal et taper la commande suivante :

$ rpicam-hello

Si cela ne fonctionne pas, c'est que la caméra n'est pas bien branché.

Installation de OpenCV :

Pour pouvoir lire les ArucoCode, l'installation de OpenCV est indispensable. Pour cela, ouvre le terminal et devra taper les commandes suivante :

$ sudo apt update && sudo apt upgrade

$ sudo apt install cmake libgtk2.0-dev

$ git clone https://github.com/opencv/opencv.git

$ mkdir opencv/build && cd opencv/build && mkdir modules

$ git clone https://github.com/opencv/opencv_contrib.git

$ mv opencv_contrib/modules/aruco modules

$ cmake -DOPENCV_EXTRA_MODULES_PATH=./modules ..

$ sudo make install

L'Installation peut prendre plusieurs heures.

Calibrage de la caméra :

Créer un dossier « Aruco » pour mettre nos codes à l'intérieur. Ouvre le terminal et tape les commandes suivantes :

$ mkdir Aruco

$ cd Aruco

$ sudo nano photo_calibrage.py

Mettre le code suivant :

```
#importation des bibliotheque
from picamera2 import Picamera2
import cv2, time

#Démarrer la cam
picam2 = Picamera2()
```

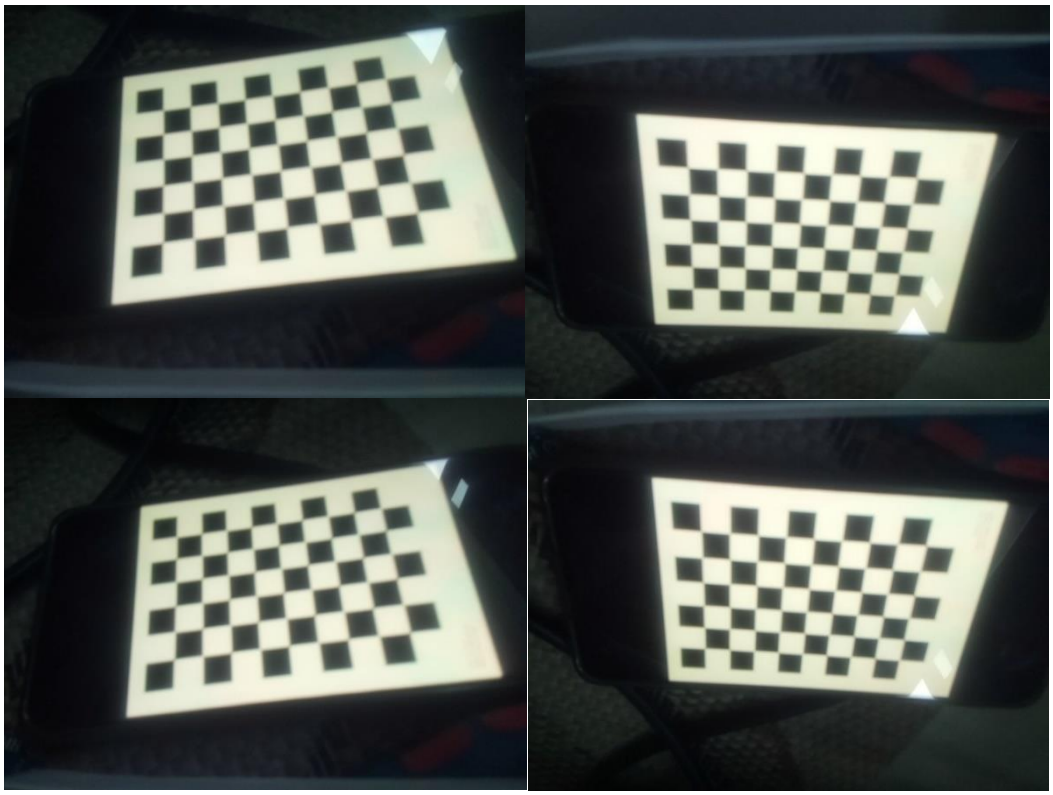
```
picam2.start()


prev_frame_time = time.time()


cal_image_count = 0
frame_count = 0


while True:
    frame = picam2.capture_array()
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    frame_count +=1
    new_frame_time = time.time()
    fps =  1/(new_frame_time - prev_frame_time)
    prev_frame_time = new_frame_time
    cv2.putText(frame, "FPS " + str(int(fps)), (10,40), cv2.FONT_HERSHEY_PLAIN, 3, (100,255,0), 2, cv2.LINE_AA)
    cv2.imshow("Camera", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cv2.destroyAllWindows()
```

Ce code permet de faire des photos à partir de la caméra. Les photos devrait être une grille qui ressemble à un grille d'échiquier sous différent angle pour pouvoir calibrer la caméra.



Puis faire un prochain code nommée calibrage.py et mettre le code suivant :

```
import numpy as np
import cv2
import glob

cd_width = 9
cd_height = 6
cd_square_size = 26.3

criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

cd_3D_points = np.zeros((cd_width * cd_height, 3), np.float32)
cd_3D_points[:,:2] = np.mgrid[0:cd_width, 0:cd_height].T.reshape(-1,2) * cd_square_size

list_cd_3d_points = []
list_cd_2d_img_points =[]
```

```
list_images = glob.glob('*.jpg')

for frame_name in list_images:
        img = cv2.imread(frame_name)

        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

        ret, corners = cv2.findChessboardCorners(gray, (9,6),None)

        if ret == True:
                list_cd_3d_points.append(cd_3D_points)

                corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
                list_cd_2d_img_points.append(corners2)

                cv2.drawChessboardCorners(img, (cd_width, cd_height), corners2,ret)
                cv2.imshow('img',img)
                cv2.waitKey(500)

cv2.destroyAllWindows()

ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(list_cd_3d_points, list_cd_2d_img_points,
gray.shape[::-1],None,None)

print("Calibration Matrix: ")
print(mtx)
print("Disortion: ",dist)

with open('camera_cal.npy','wb') as f:
        np.save(f, mtx)
        np.save(f, dist)
```

Ce code va créer un code camera_cal.npy qui va être utilisé pour la détection des Aruco.

Detection des Aruco:

Enfin voici le code pour le ArucoCode. Ce code permet de détecter les aruco et donne son numéro :

```
import numpy as np
import cv2
import cv2.aruco as aruco
from picamera2 import Picamera2
import time

marker_size = 100
with open('camera_cal.npy', 'rb') as f:
        camera_matrix = np.load(f)
        camera_distortion = np.load(f)

aruco_dict = aruco.getPredefinedDictionary(aruco.DICT_4X4_250)

picam2 = Picamera2()
picam2.start()

while True:

        frame = picam2.capture_array()

        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        corners, ids, rejected = aruco.detectMarkers(gray_frame, aruco_dict, camera_matrix,
camera_distortion)
```

```
        if ids is not None:

                aruco.drawDetectedMarkers(frame, corners)

                rvec, tvec, _objPoints = aruco.estimatePoseSingleMarkers(corners, marker_size,
camera_matrix, camera_distortion)

                for marker in range(len(ids)):
                        cv2.drawFrameAxes(frame, camera_matrix, camera_distortion, rvec[marker],
tvec[marker], 100)
                        cv2.putText(frame, str(ids[marker][0]) ,( int(corners[marker][0][0][0]) - 30,
int(corners[marker][0][0][1])), cv2.FONT_HERSHEY_PLAIN, 3, (255, 0, 0), 2, cv2.LINE_AA)



        cv2.imshow('frame', frame)

        key = cv2.waitKey(1) & 0xFF
        if key == ord('q'): break

cv2.destroyAllWindows()
```