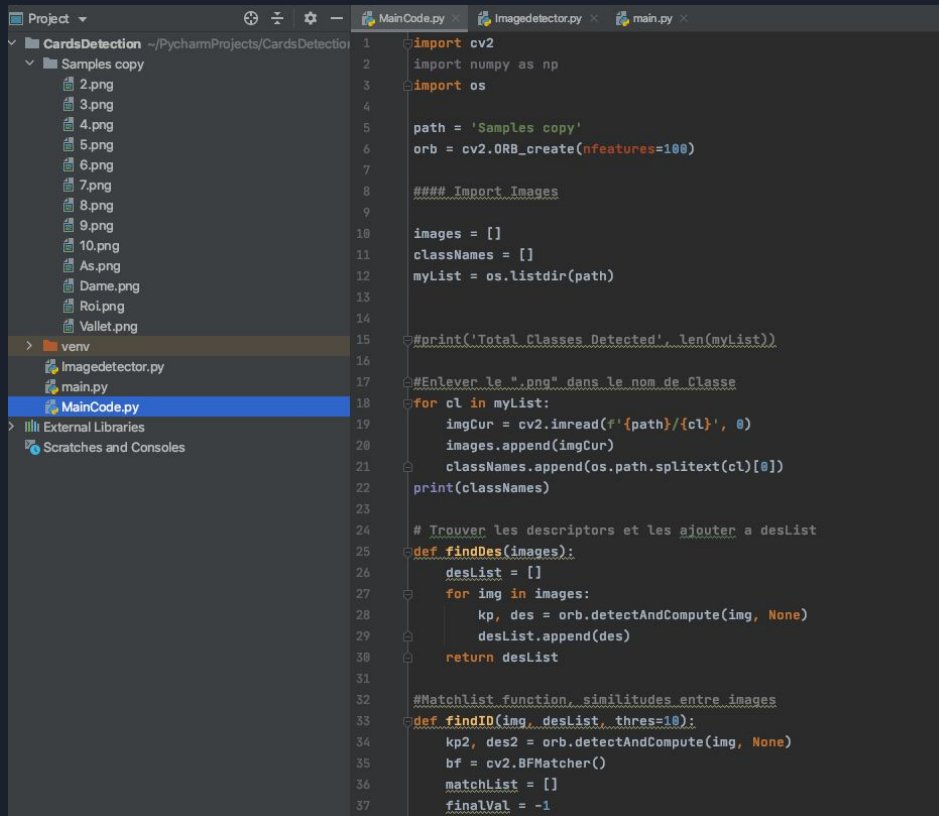


Partie Algorithmie :

- PyCharm
- OpenCV
- Programmation orientée objet



The screenshot shows the PyCharm IDE interface. On the left, the 'Project' view displays a folder named 'CardsDetection' located at '~\PycharmProjects\CardsDetection'. Inside this folder is a sub-folder 'Samples copy' containing several image files: 2.png, 3.png, 4.png, 5.png, 6.png, 7.png, 8.png, 9.png, 10.png, As.png, Dame.png, Roi.png, and Valet.png. Below the 'Samples copy' folder is a 'venv' folder, and further down are files 'Imagedetector.py' and 'main.py'. The 'MainCode.py' file is currently selected and open in the editor. The code in 'MainCode.py' is as follows:

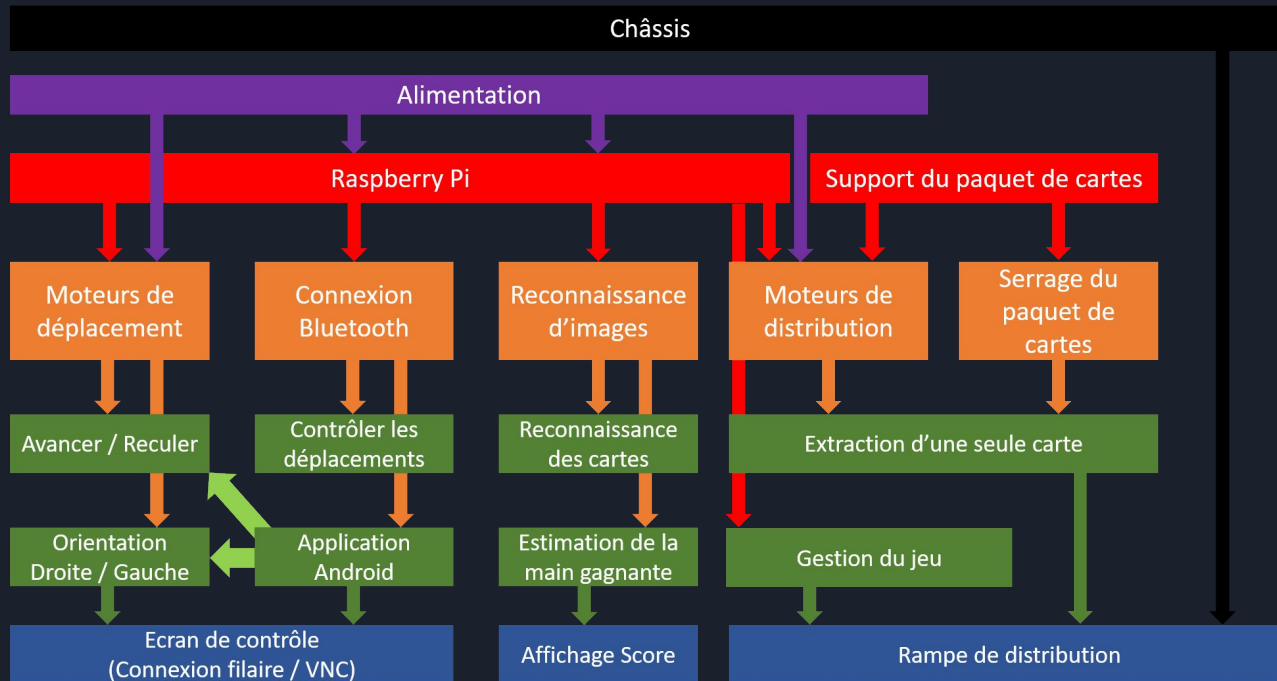
```
1 import cv2
2 import numpy as np
3 import os
4
5 path = 'Samples copy'
6 orb = cv2.ORB_create(nfeatures=100)
7
8 ##### Import Images
9
10 images = []
11 classNames = []
12 myList = os.listdir(path)
13
14
15 #print('Total Classes Detected', len(myList))
16
17 #Enlever le ".png" dans le nom de Classe
18 for cl in myList:
19     imgCur = cv2.imread(f'{path}/{cl}', 0)
20     images.append(imgCur)
21     classNames.append(os.path.splitext(cl)[0])
22     print(classNames)
23
24 # Trouver les descriptors et les ajouter a desList
25 def findDes(images):
26     desList = []
27     for img in images:
28         kp, des = orb.detectAndCompute(img, None)
29         desList.append(des)
30     return desList
31
32 #Matchlist function, similitudes entre images
33 def findID(img, desList, thres=10):
34     kp2, des2 = orb.detectAndCompute(img, None)
35     bf = cv2.BFMatcher()
36     matchList = []
37     finalVal = -1
```



Soutenance avancement Projet ER3 : Robot Croupier

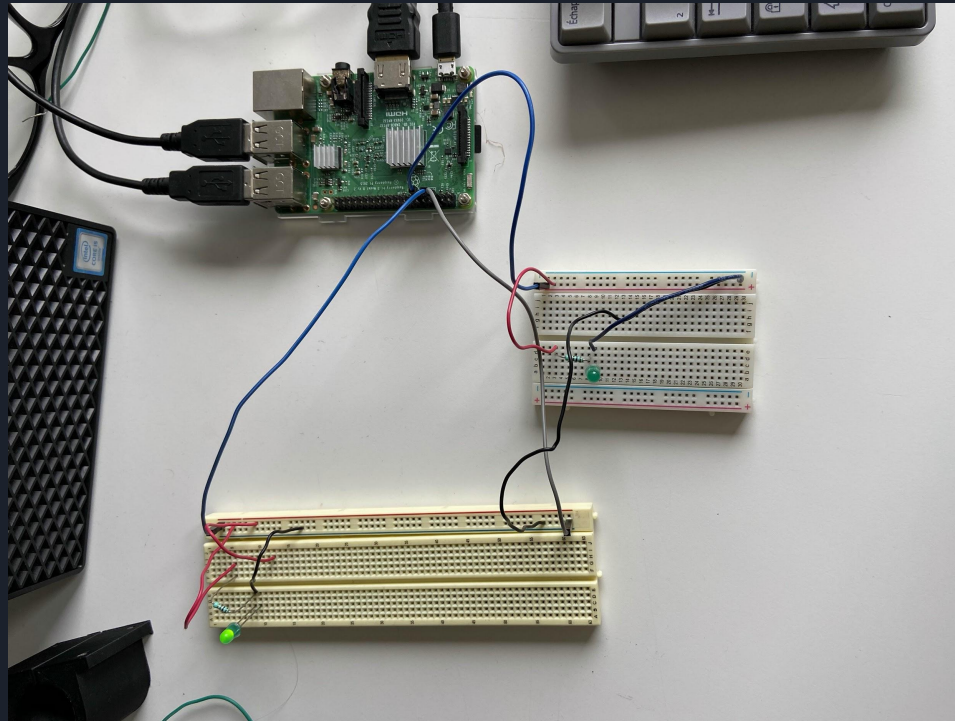
Batiste Laloi
Vahé Djorkaeff
307


Partie Moteurs :



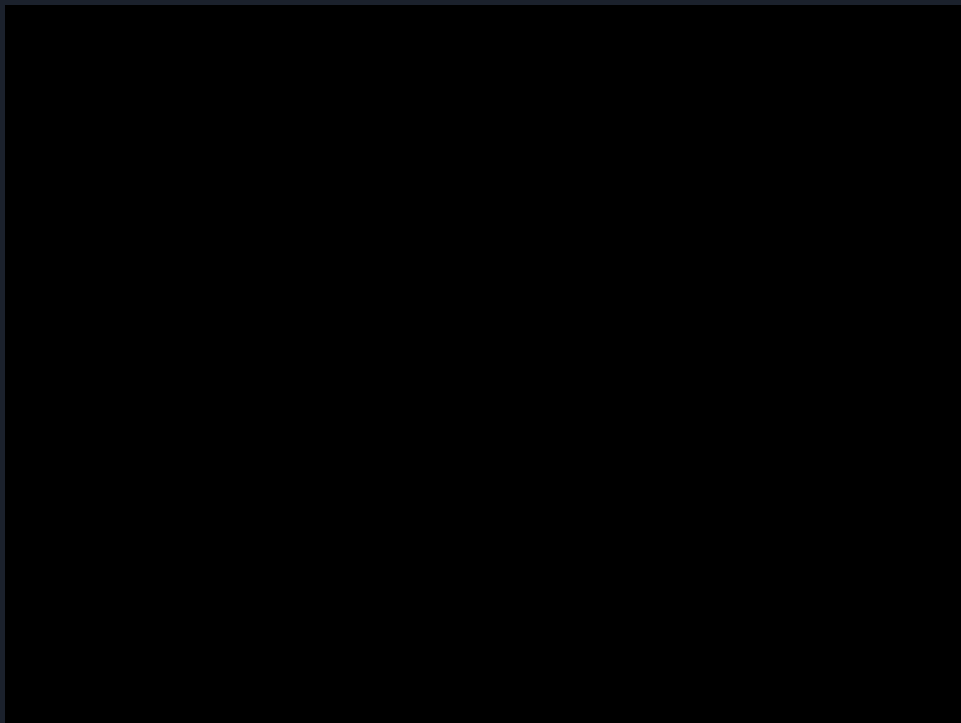
- Connexion bluetooth
- Motricité des 4 moteurs

Schéma du montage permettant de commander deux LEDs avec notre Raspberry Pi reliée en bluetooth à une télécommande sur notre téléphone.

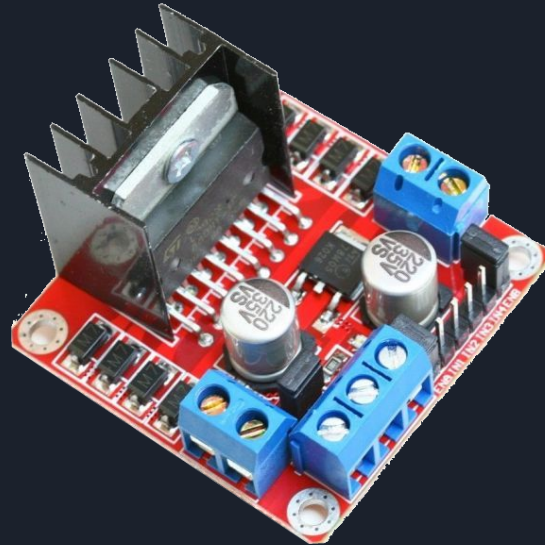




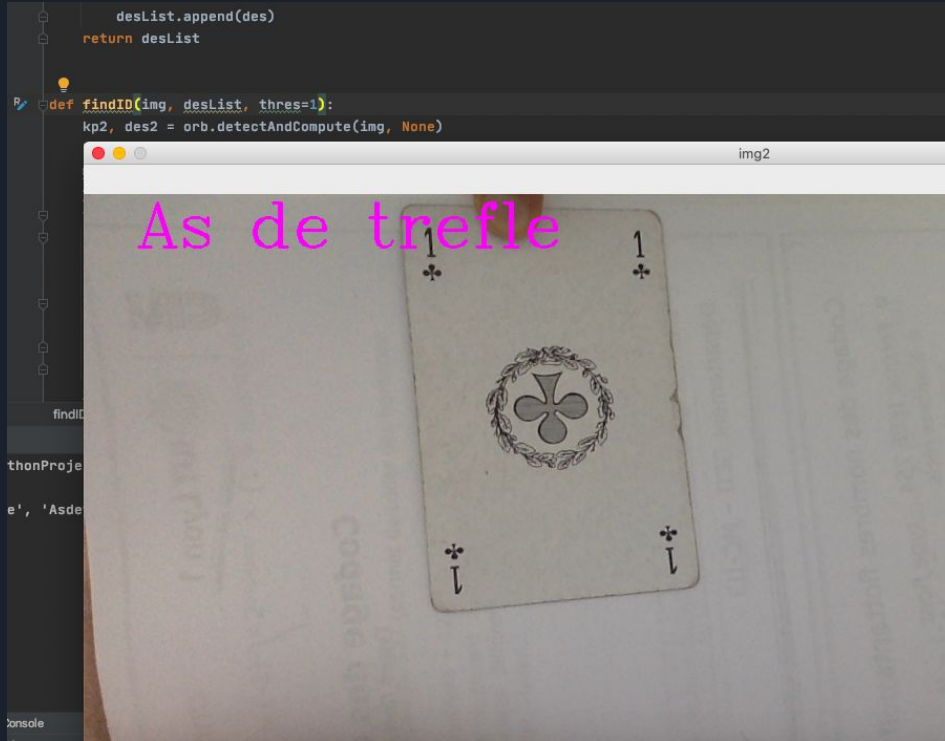
Vidéo du montage permettant de commander 4 LEDs
représentant les 4 roues de notre robot croupier.



Malheureusement nous n'avons pas encore eu l'occasion de tester ce programme fonctionnant avec les LEDs sur les moteurs fixés au châssis que nous possédons car il nous manque un double pont en H. Il sert à sécuriser les broches de la Raspberry Pi afin que les courant intenses parcourant les moteurs de les endommagent pas.



Détection de cartes :



Nous avons réussi la partie détection de cartes mais il reste encore quelques points à optimiser pour améliorer le temps d'exécution du programme.

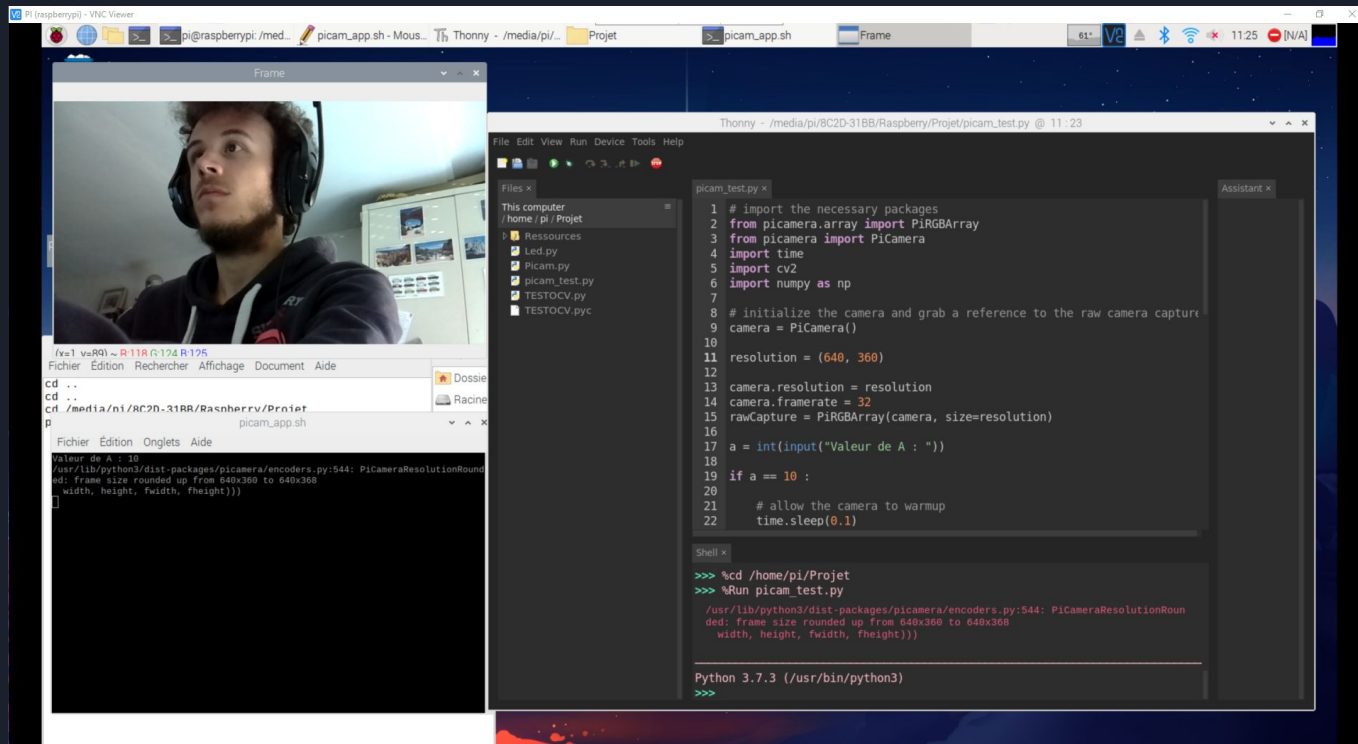


Traitement d'image :

Nous utiliserons notre caméra branchée à la Raspberry Pi et OpenCV qui est une bibliothèque graphique Python permettant de travailler sur la reconnaissance d'images et donc ici, qui va nous servir à reconnaître les cartes distribuées pas le robot.



Traitement d'image :



Programmation Orientée Objet :

```
24
25 #   Creation de la classe Robot
26 #   Affectation des arguments des objets liés à cette classe
27 #   Self -->"Self affectation"
28 class Carte:
29     def __init__(self, valeur, couleur, position):
30         self.valeur = valeur
31         self.couleur = couleur
32         self.position = position
33
34     # Creation des "methodes" = Fonctions propres
35     # OBJET.fonction --> execution de la routine
36     #   def change_pos(self):
37
38 class Joueur:
39     def __init__(self, carte_1, carte_2, debout, cote):
40         self.carte_1 = carte_1
41         self.carte_2 = carte_2
42         self.debout = debout
43         self.cote = cote
44
45     #   def Affectation_main(self):
46
```

