Plan pour le POC

Concernant le POC, nous avons décidé de réaliser le programme permettant de faire fonctionner la borne. Pour cela nous avons décidé d'utiliser un Raspberry Pi. Cela nous a permis d'avoir un nano-ordinateur dans lequel nous avons accès à Python, notre logiciel de programmation.

Il nous a fallu d'abord comprendre le Raspberry Pi puisqu'il s'agissait de la première fois que nous en avions un en main. Une fois les périphériques branchés, nous avons commencé à imaginer le code. Le premier problème que nous avons eu a été de trouver la bonne bibliothèque permettant à la fois de faire en sorte que le programme détecte la caméra, mais en plus qu'il détecte des code-barres et nous donne enfin son code. Nous n'avions pas connaissance d'une bibliothèque Python qui nous permettrait de faire toutes ces étapes, il nous a donc fallu chercher des bibliothèques sur internet. Nous en avons essayé plusieurs comme "Pyzbar" ou "OpenCv" mais elles n'ont mené à rien car elles étaient soit trop complexes à mettre en place, par exemple, une étape demandée pour mettre en place OpenCv était de modifier le code natif de Python via son langage codé en C ce qui était trop difficile à faire pour nous, soit elles ne correspondaient pas à ce que nous voulions vraiment, par exemple, "Pyzbar" ne nous permettait que de détecter un code-barre sur une image ce qui n'est pas vraiment ce que nous voulions.

Finalement nous avons trouvé la bibliothèque "zbarcam" qui permet de faire en sorte que notre caméra détecte des code-barres, les scanne et nous renvoie son code. Le seul problème a été que zbarcam s'utilise sur le shell sous Linux et non dans la console Python directement. Nous avons donc utilisé une autre bibliothèque qui est "subprocess" permettant à partir du code python permet de faire des commandes shell sans passer par l'invité de commandes. On a utilisé la méthode run() pour le faire et la méthode stdout() pour récupérer le code-barre scanné par notre caméra. Enfin nous avons fait une boucle 'if' pour que quand le code-barre d'un de nos articles que nous avons utilisé, il puisse nous afficher une image qui nous dit dans quelle poubelle on doit jeter l'article (on a utilisé la méthode run() de subprocess avec la commande "display" du shell)

Cette boucle 'if' est une bonne méthode pour un nombre de code-barres limité comme pour notre POC (quelques articles). A moyenne échelle (avec des milliers d'articles), le programme serait très long à mettre en place avec toutes ces boucles 'if". On pourrait utiliser un tableur Excel afin de renseigner les code-barres et les poubelles correspondantes et faire en sorte que le programme Python lise le tableur jusqu'à trouver le code-barres. A très grande échelle (millions d'articles), le tableur ne serait plus suffisant, Python mettrait trop de temps à retrouver le code-barres. À cette échelle, il faudrait utiliser une vraie base de données sous SQL par exemple.