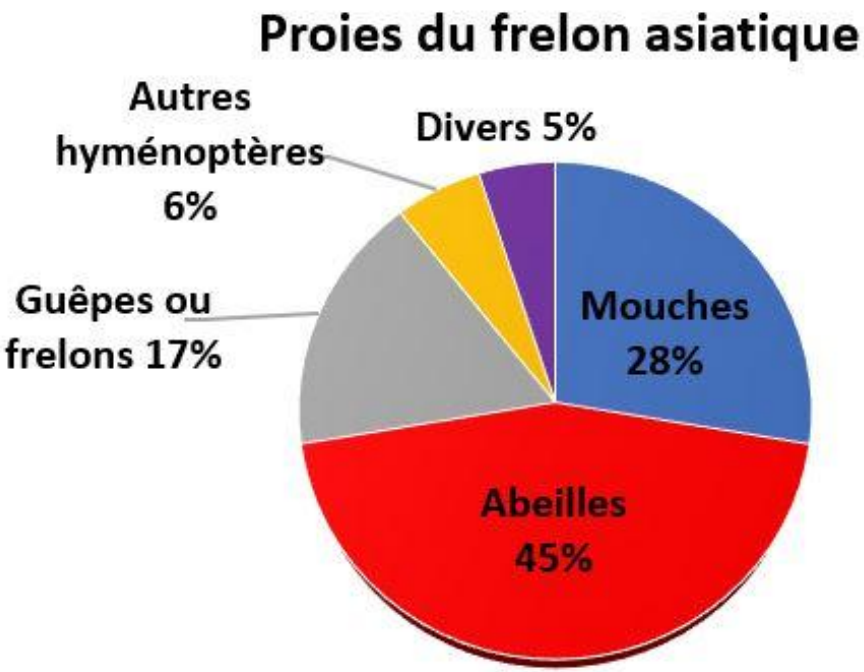
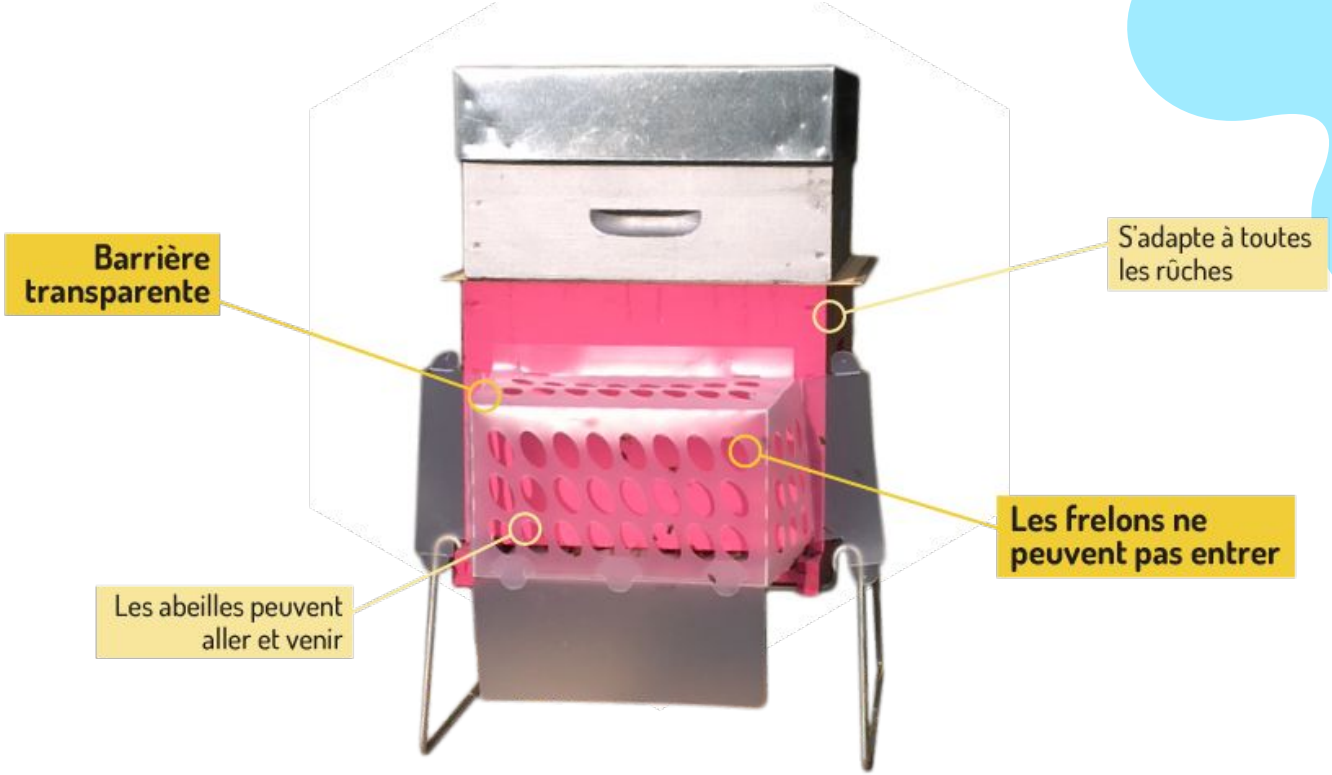
Projet KYUH

Session 2021

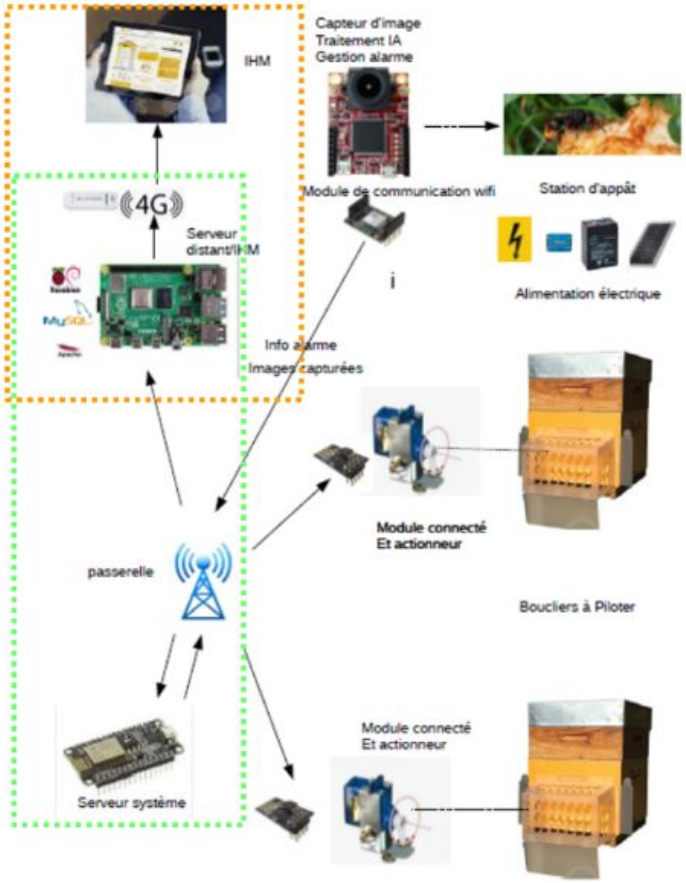
Kilian FERREIRA Yanis HADJ-BELGACEM Hugo Arbona Ugo Acquati

Expression du besoin


Proies du frelon asiatique
Autres hyménoptères 6%
Divers 5%
Guêpes ou frelons 17%
Mouches 28%
Abeilles 45%

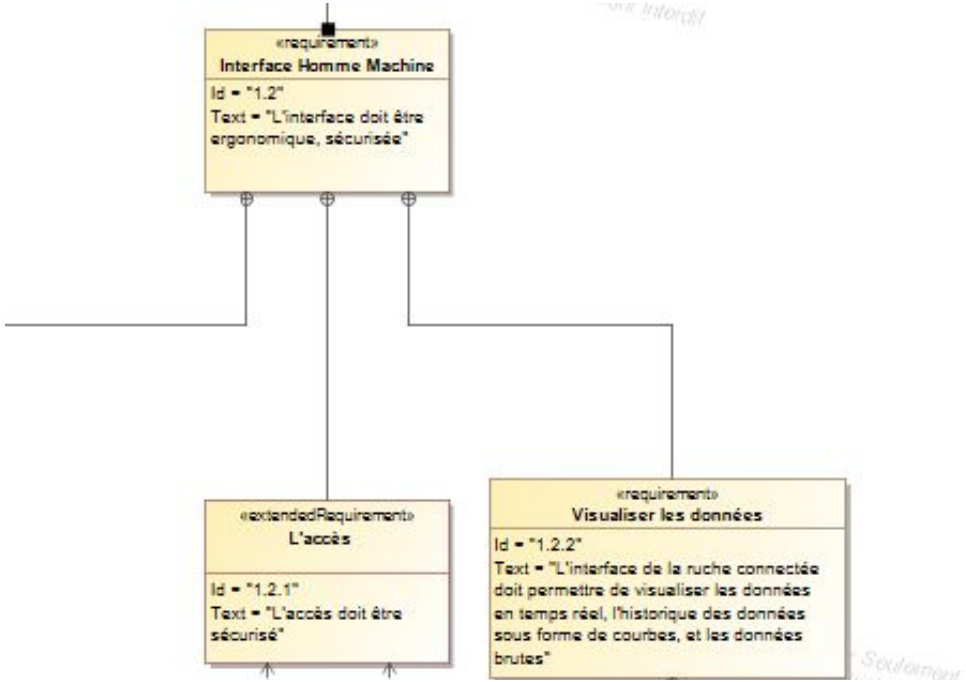

Barrière transparente
S'adapte à toutes les rûches
Les frelons ne peuvent pas entrer
Les abeilles peuvent aller et venir

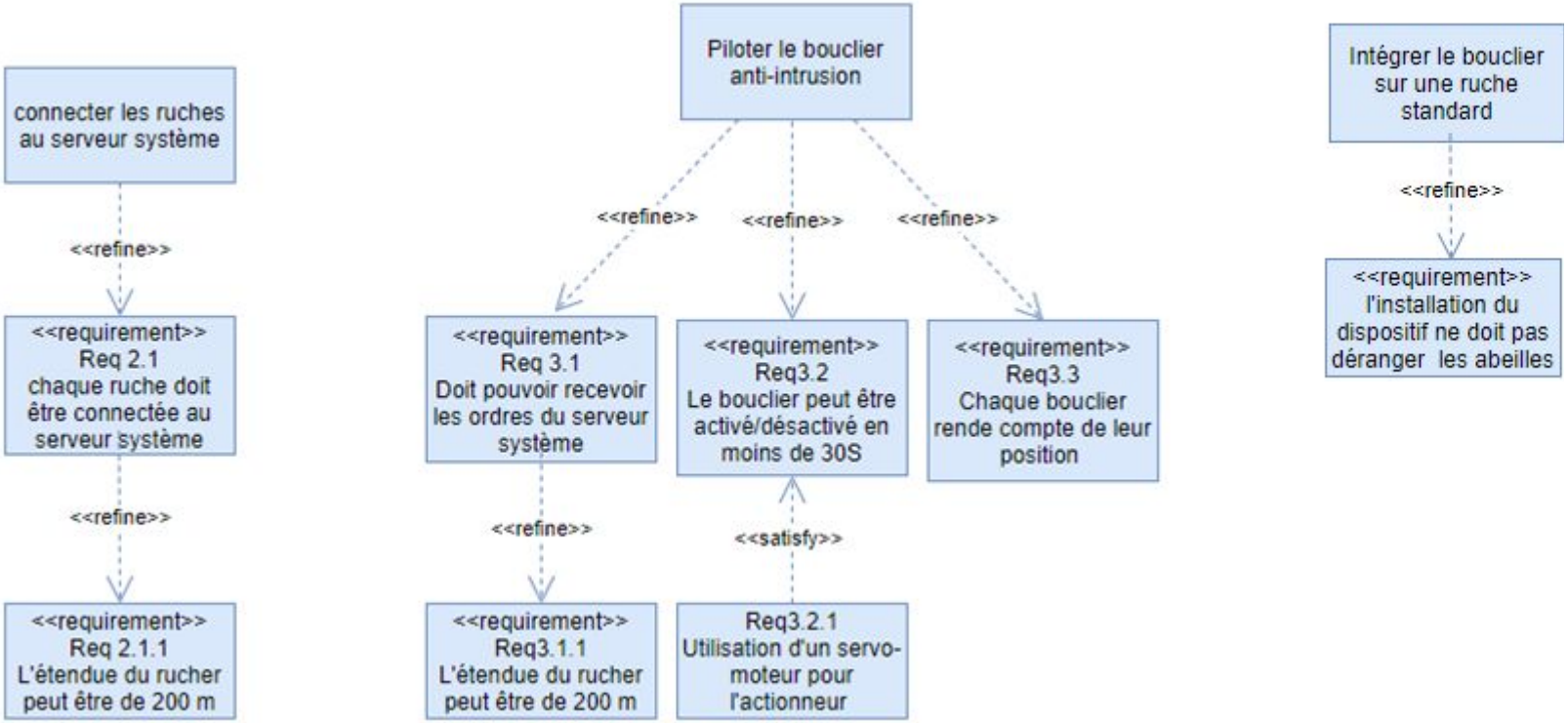
Présentation du projet

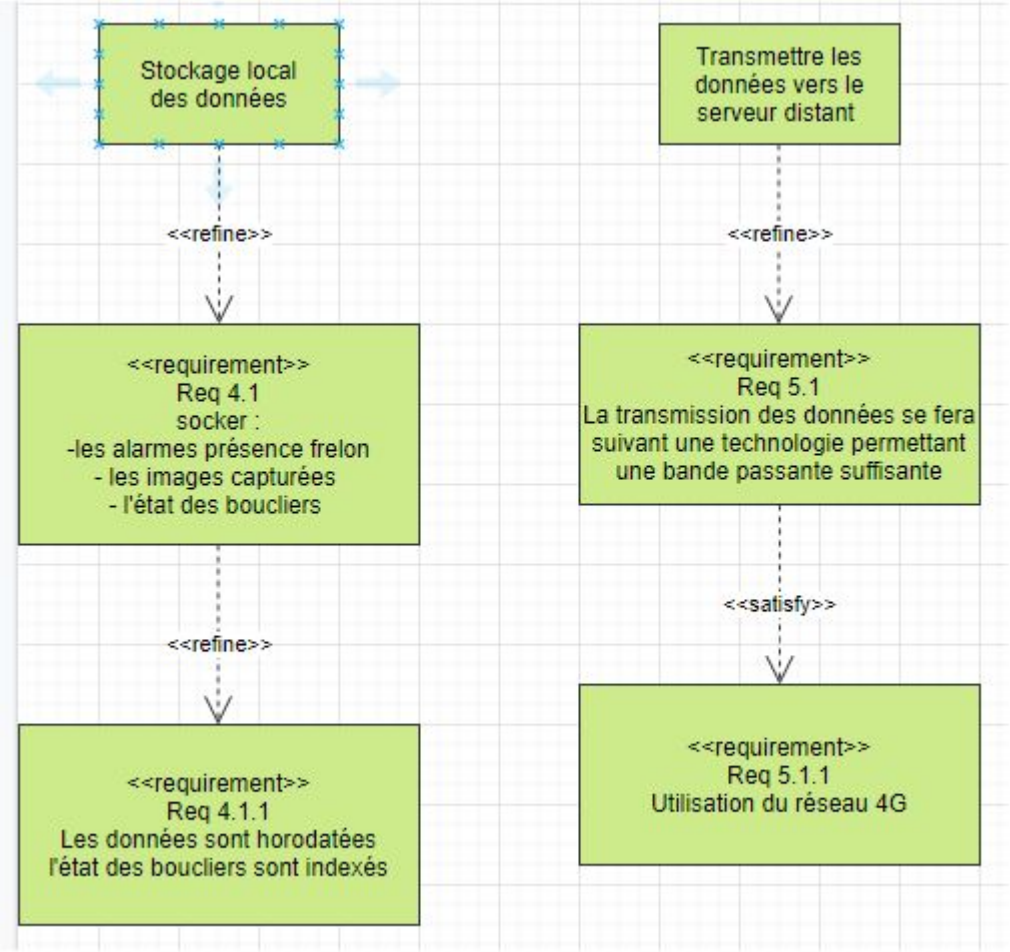


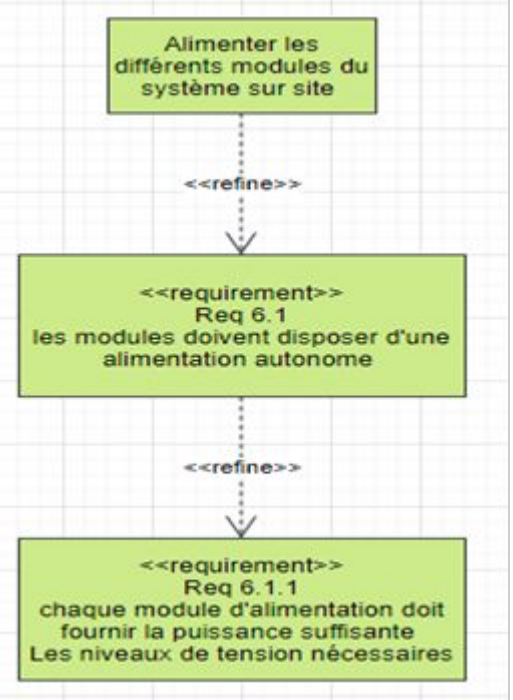
Répartition des tâches

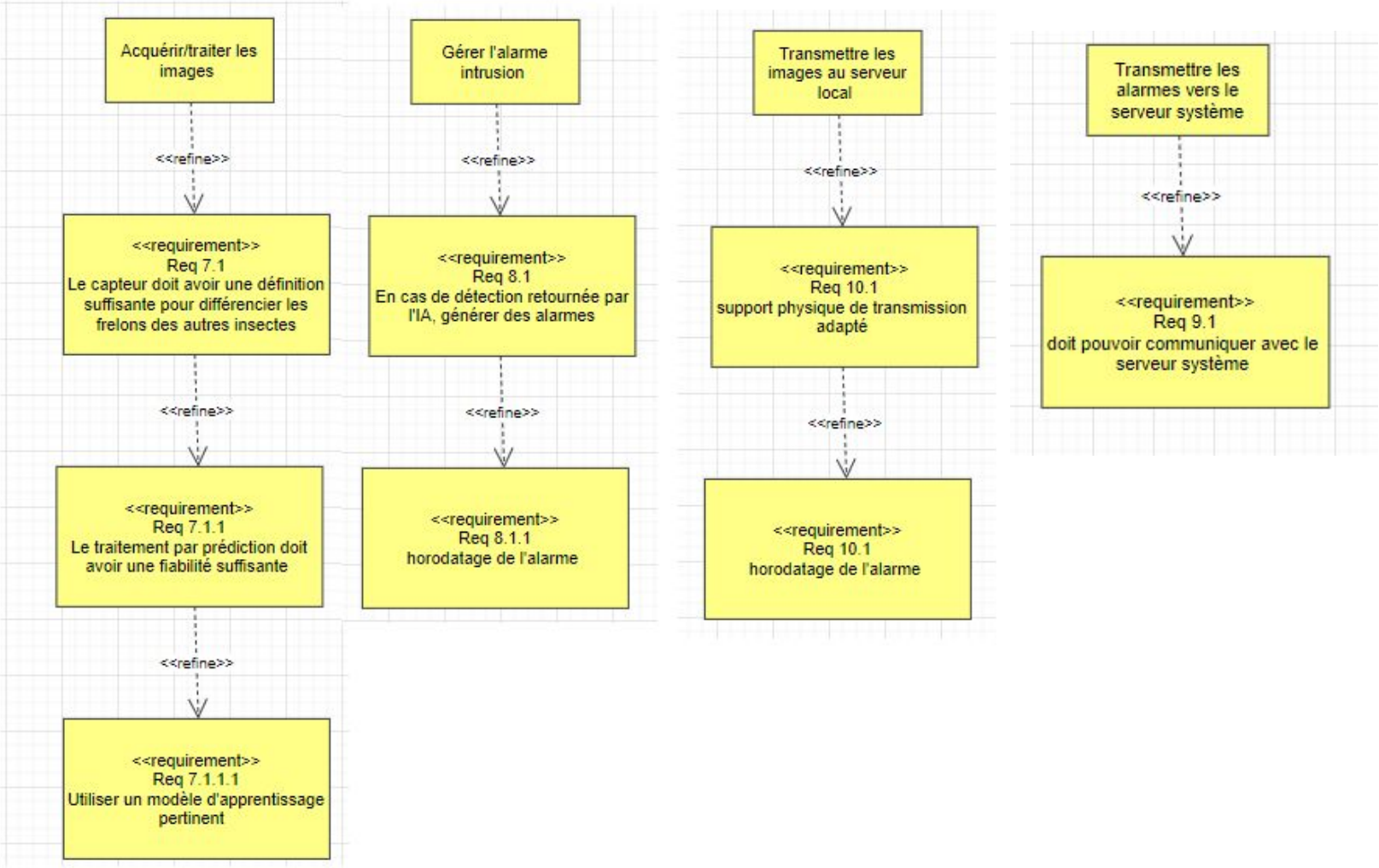
Exigences


«requirement»
Interface Homme Machine
Id = "1.2"
Text = "L'interface doit être ergonomique, sécurisée"
«extendedRequirement»
L'accès
Id = "1.2.1"
Text = "L'accès doit être sécurisé"
«requirement»
Visualiser les données
Id = "1.2.2"
Text = "L'interface de la ruche connectée doit permettre de visualiser les données en temps réel, l'historique des données sous forme de courbes, et les données brutes"

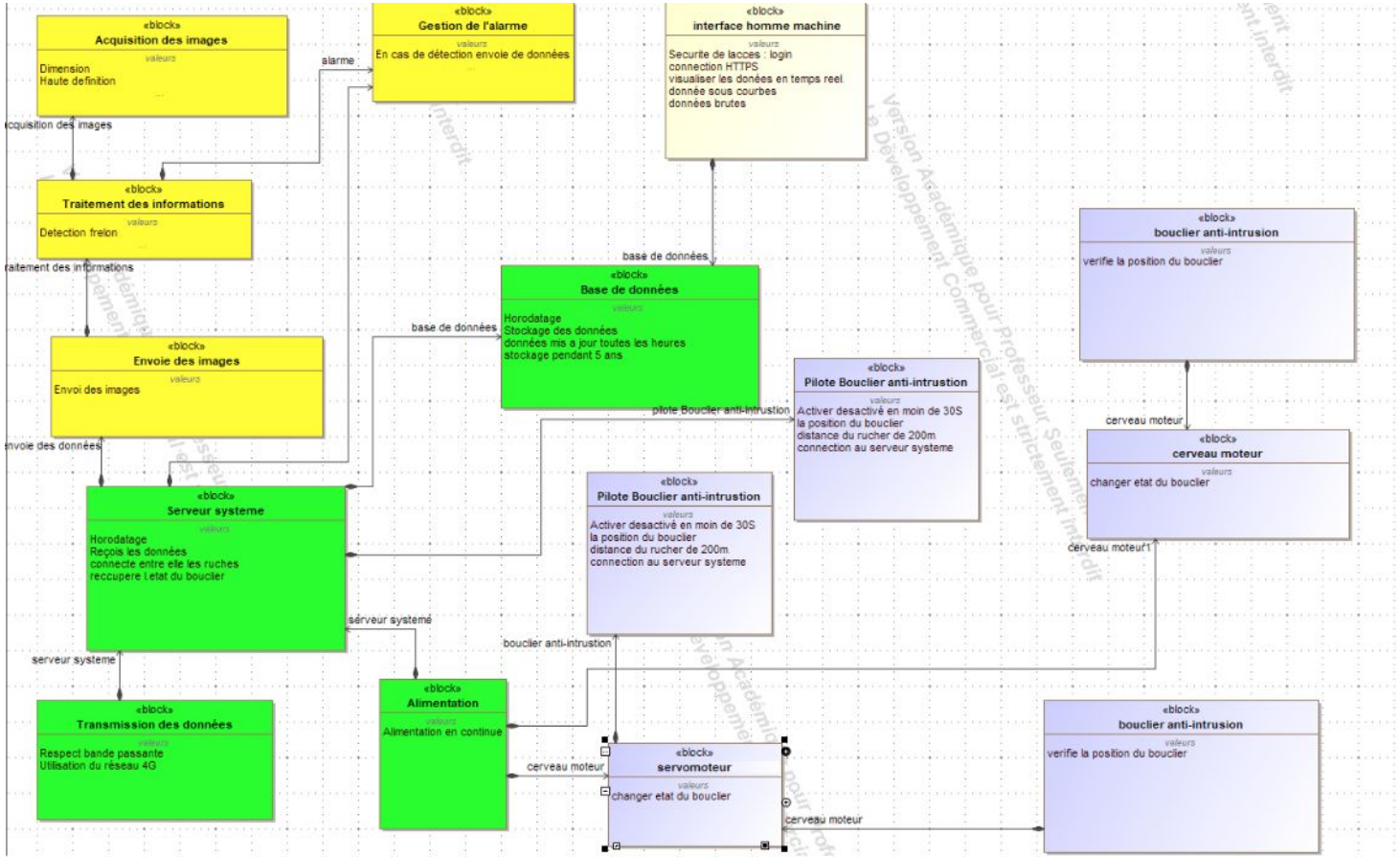

connecter les ruches au serveur système
<<refine>>
<<requirement>> Req 2.1 chaque ruche doit être connectée au serveur système
<<refine>>
<<requirement>> Req 2.1.1 L'étendue du rucher peut être de 200 m
Piloter le bouclier anti-intrusion
<<refine>>
<<refine>>
<<refine>>
<<requirement>> Req 3.1 Doit pouvoir recevoir les ordres du serveur système
<<requirement>> Req3.2 Le bouclier peut être activé/désactivé en moins de 30S
<<requirement>> Req3.3 Chaque bouclier rende compte de leur position
<<refine>>
<<satisfy>>
<<requirement>> Req3.1.1 L'étendue du rucher peut être de 200 m
Req3.2.1 Utilisation d'un servo-moteur pour l'actionneur
Intégrer le bouclier sur une ruche standard
<<refine>>
<<requirement>> l'installation du dispositif ne doit pas déranger les abeilles


Stockage local des données
<<refine>>
<<requirement>>
Req 4.1
socker :
-les alarmes présence frelon
- les images capturées
- l'état des boucliers
<<refine>>
<<requirement>>
Req 4.1.1
Les données sont horodatées
l'état des boucliers sont indexés
Transmettre les données vers le serveur distant
<<refine>>
<<requirement>>
Req 5.1
La transmission des données se fera suivant une technologie permettant une bande passante suffisante
<<satisfy>>
<<requirement>>
Req 5.1.1
Utilisation du réseau 4G

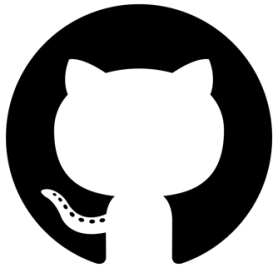

Alimenter les différents modules du système sur site
<<refine>>
<<requirement>>
Req 6.1
les modules doivent disposer d'une alimentation autonome
<<refine>>
<<requirement>>
Req 6.1.1
chaque module d'alimentation doit fournir la puissance suffisante
Les niveaux de tension nécessaires


Acquérir/traiter les images
<<refine>>
<<requirement>>
Req 7.1
Le capteur doit avoir une définition suffisante pour différencier les frelons des autres insectes
<<refine>>
<<requirement>>
Req 7.1.1
Le traitement par prédiction doit avoir une fiabilité suffisante
<<refine>>
<<requirement>>
Req 7.1.1.1
Utiliser un modèle d'apprentissage pertinent
Gérer l'alarme intrusion
<<refine>>
<<requirement>>
Req 8.1
En cas de détection retournée par l'IA, générer des alarmes
<<refine>>
<<requirement>>
Req 8.1.1
horodatage de l'alarme
Transmettre les images au serveur local
<<refine>>
<<requirement>>
Req 10.1
support physique de transmission adapté
<<refine>>
<<requirement>>
Req 10.1
horodatage de l'alarme
Transmettre les alarmes vers le serveur système
<<refine>>
<<requirement>>
Req 9.1
doit pouvoir communiquer avec le serveur système

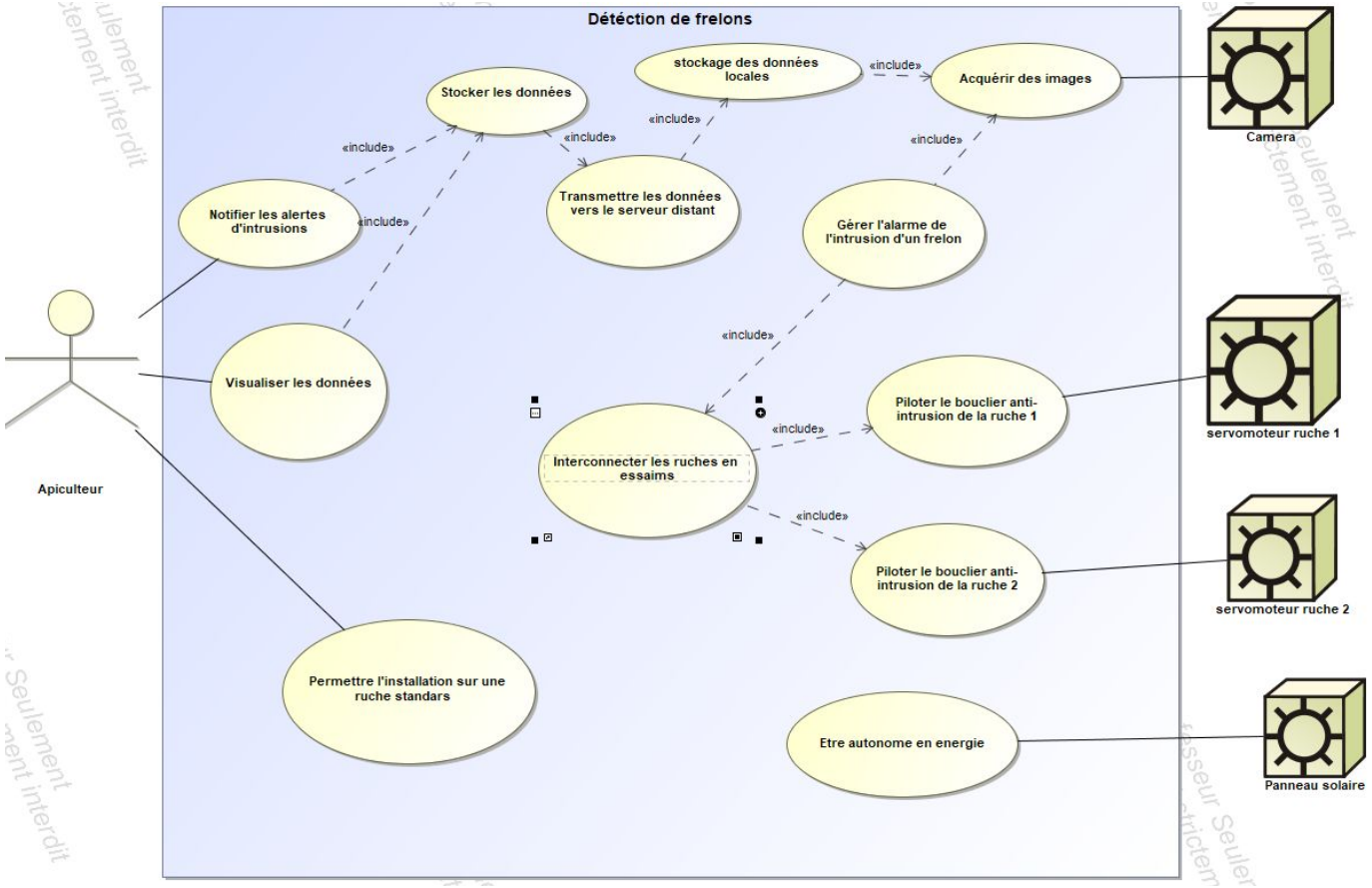
Architecture du système



Les ressources de développement

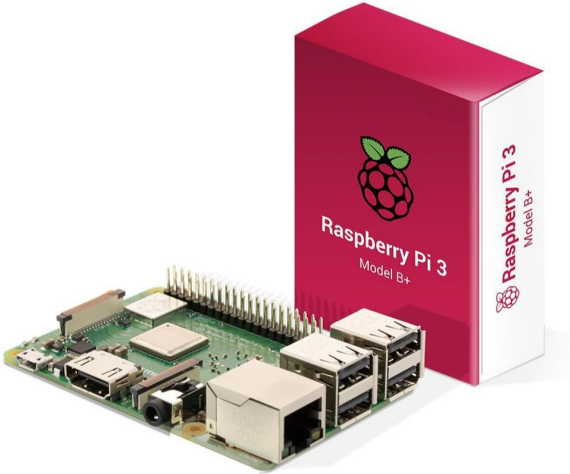


Analyse du projet



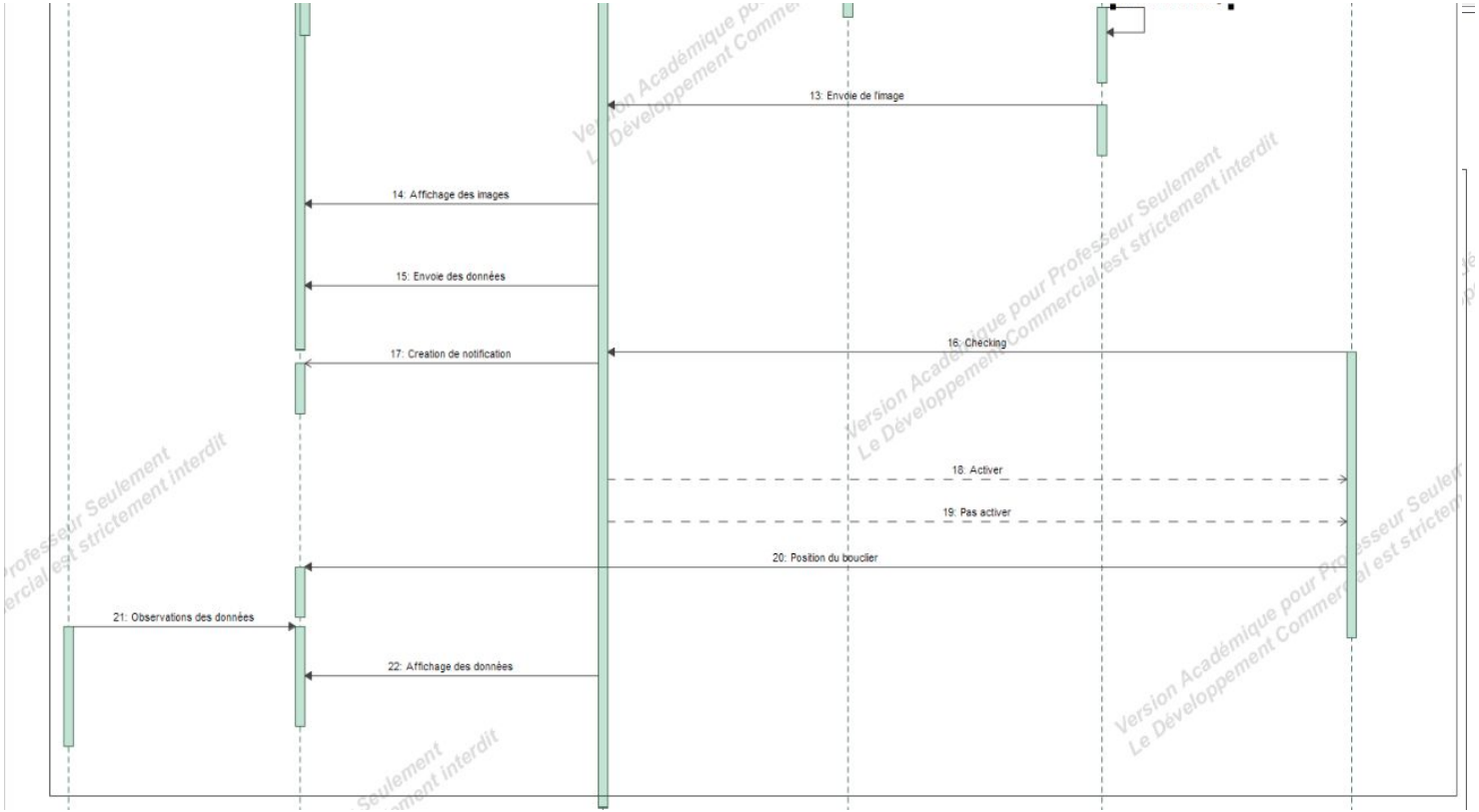
Architecture logicielle

Base de données

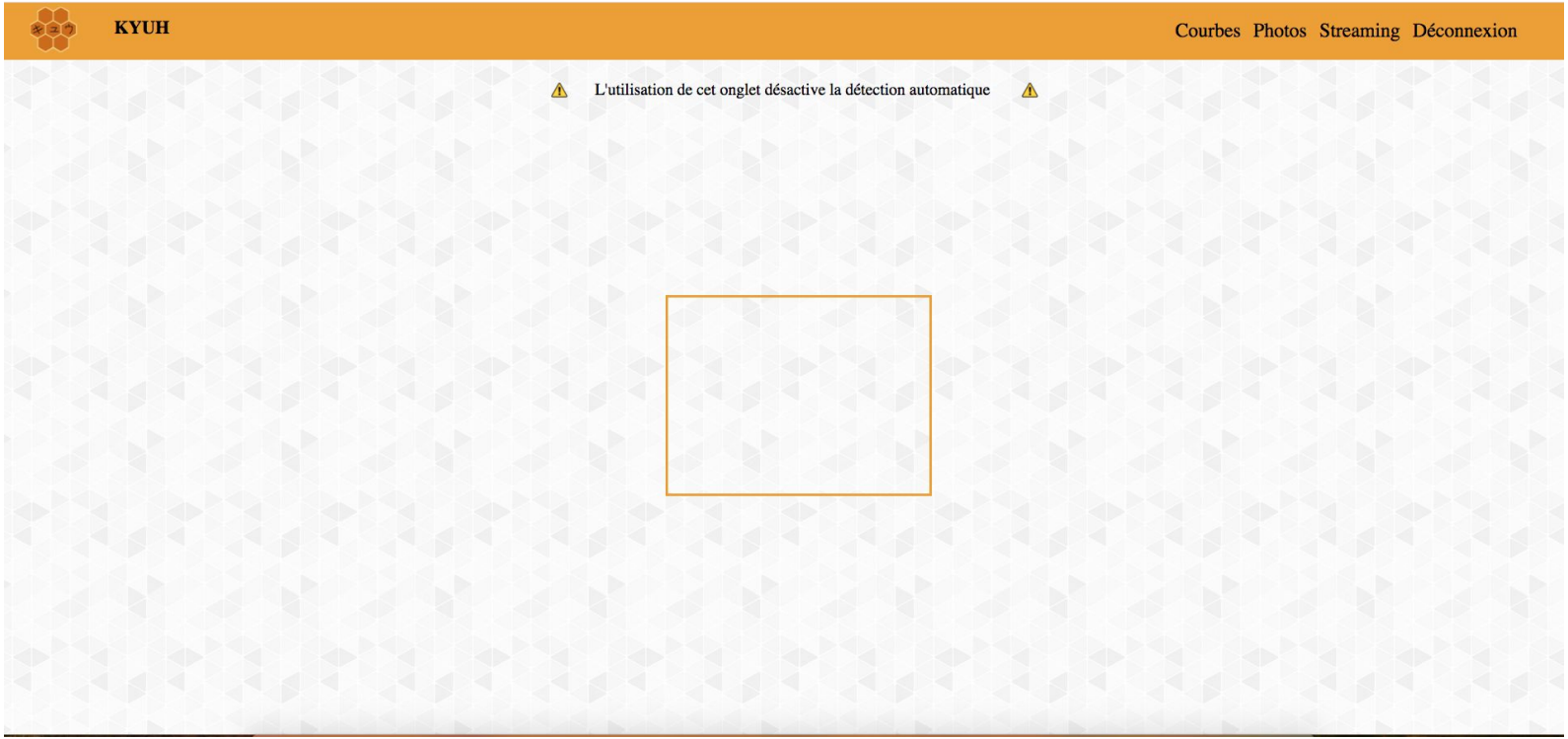

Raspberry Pi 3
Model B+
Raspberry Pi 3
Model B+

OpenMV

Scénario

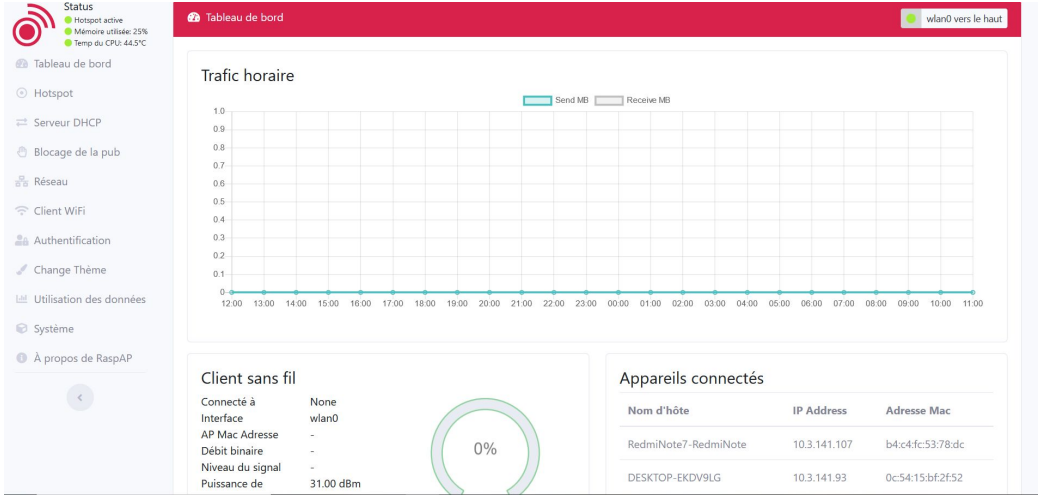

13: Envoie de l'image
14: Affichage des images
15: Envoie des données
16: Checking
17: Creation de notification
18: Activer
19: Pas activer
20: Position du bouclier
21: Observations des données
22: Affichage des données

Test unitaire



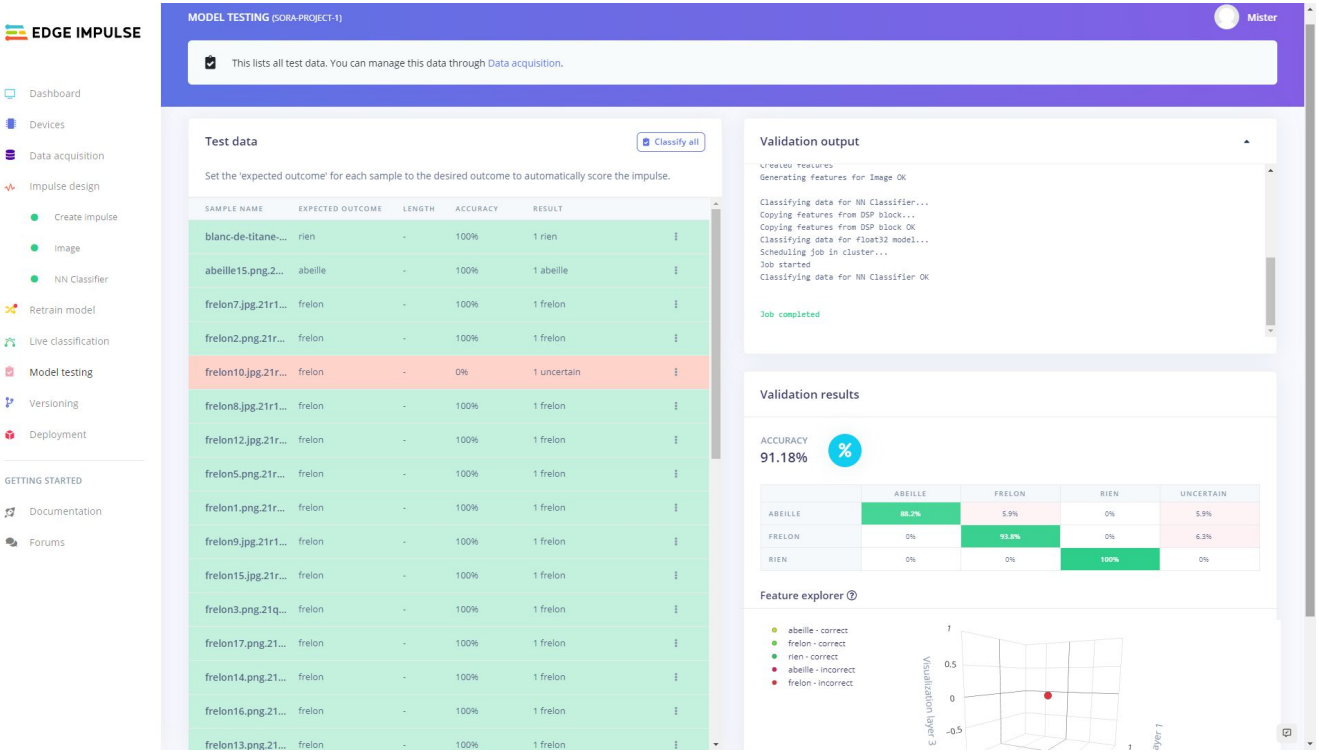
```
Envoi d'une requête 'Ping'  10.3.141.107 avec 32 octets de données :
Réponse de 10.3.141.107 : octets=32 temps=362 ms TTL=64
Réponse de 10.3.141.107 : octets=32 temps=132 ms TTL=64
Réponse de 10.3.141.107 : octets=32 temps=259 ms TTL=64
Réponse de 10.3.141.107 : octets=32 temps=36 ms TTL=64

Statistiques Ping pour 10.3.141.107:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 36ms, Maximum = 362ms, Moyenne = 197ms
```



Pilote bouclier

Création d’un model



Scénario(s)

```
while(True):
    addr = socket.getaddrinfo('10.3.141.1', 80)[0][-1]
    client = socket.socket()
    try:
        client.connect(addr)
        conn = 1
    except:
        conn = 0
    if conn == 1:
        clock.tick()
        compteur=0
        fiable=0
        while(compteur!=10):
            img = sensor.snapshot()
            for obj in tf.classify(net, img, min_scale=1.0, scale_mul=0.8, x_overlap=0.5, y_overlap=0.5):
                print("**********\nPredictions at [x=%d,y=%d,w=%d,h=%d]" % obj.rect())
                img.draw_rectangle(obj.rect())
                predictions_list = list(zip(labels, obj.output()))

                for i in range(len(predictions_list)):
                    print("%s = %f" % (predictions_list[i][0], predictions_list[i][1]))

                if predictions_list[1][1] >= 0.6 :
                    print('fiable+1')
                    fiable+=1
                else:
                    print('fiable-1')
                    fiable-=1
            print(clock.fps(), "fps")
            compteur+=1
        print("\n\n\n fiabilité :")
        print(fiable)
        if fiable>=0 :
            sensor.snapshot().save("example.jpg")
            files = open('example.jpg', 'rb')
            client.write("POST /update HTTP/1.0\r\n")
            client.write("Host: " + HOST + "\r\n")
            client.write("Content-Length: \r\n\r\n")
            client.write(files)
            print(client.read(128))
            print("send")
          client.close()
```

```
"Host: 10.
"Content-T
"Cache-Con
"Pragma: no

client.close()
```

```
fiabilité :
send

Traceback (most recent call last):
  File "<stdin>", line 79, in <module>
OSError: -6
MicroPython: v1.13-r84 OpenMV: v3.9.4 HAL
Type "help()" for more information.
>>>
```

Merci