Cuadros Alexandre

Valentin Taffin

IMA 4

RAPPORT DE PROJET

« Les mini drones»

Années 2015-2016

REMERCIEMENTS

Nous remercions tout particulièrement notre tuteur Mr Redon pour nous avoir permis de travailler sur ce projet intéressant et de nous avoir fourni le matériel nécessaire.

INTRODUCTION

La robotique comme l'a prédit Isaac Asimov est présente dans tous les domaines et a le pouvoir de regrouper la plupart des sciences humaines comme l'informatique et l'électronique, tout en restant un moyen ludo-éducatif. Dans notre cas la société Parrot a développé plusieurs drones, et nous avons la chance à travers ce projet d'en utiliser deux : un roulant et un volant.

Nous avons à notre disposition deux drones conçus par la société Parrot.

La Jumping Sumo :

Ce drone a la capacité de se déplacer en roulant puis de sauter. Celui-ci sera commandé par wifi.

Nous avons ensuite la Rolling Spider :

Ce drone a la possibilité de se déplacer en volant et possède des roues protectrices en cas de chute. Celui-ci sera commandé par Bluetooth.

Le but de notre projet est donc de coordonner les deux drones pour qu'ils puissent se suivre. Pour ce faire, la Rolling Spider analyse les images reçues en suivant la Jumping Sumo de dessus. L'image devra être respectée, c'est à dire que la position de la Jumping Sumo devra être identique, si ce n'est pas le cas, la Rolling Spider interagira en fonction de la position du drone roulant sur l'image précédente et rectifiera l'image. Nous utiliserons dans ce cas la programmation en Open CV.

SOMMAIRE

I) CARACTERISTIQUES DES DRONES :

Voici ci-dessous les caractéristiques techniques de nos deux drones :

Caractéristiques	RollingSpider	JumpingSumo
Portée	Jusqu'à 20 mètres.	Jusqu'à 50 mètres.
Connectivité	Technologie Bluetooth Smart, V4.0 BLE (Bluetooth Low Energy).	Génère son réseau WiFi® (AC) 2.4 ou 5 Ghz.
Stabilité	Un capteur ultrasons,un gyroscope 3 axes et d'un accéléromètre 3 axes.	Centrale inertielle avec un gyroscope et un accéléromètre.
Batterie	Lithium-Polymer et amovible. Autonomie de 8 minutes et recharge complète en 90 minutes.	Lithium-Polymer et amovible. Autonomie de 20 minutes et recharge complète en 90 minutes.
Indicateur	Des LED bicolores indiquent le statut de la Rolling Spider.	Des yeux lumineux indiquent le statut du Jumping Sumo.
Caméra	Une caméra verticale et d'un capteur de pression.	Le flux est retransmis, en streaming, sur l'écran du smartphone ou tablette de pilotage Résolution : 640 x 480 px, 15 images par seconde.
Vitesse		Roule à 2 m/s (7 km/h).

II) INSTALLATION ET CONFIGURATION :

1) FreeFlight 3 :

Pour débuter, nous avons essayé l'application gratuite FreeFlight 3 disponible sur Android :

Cette application permet de contrôler avec simplicité les drones à l'aide de notre Smartphone.

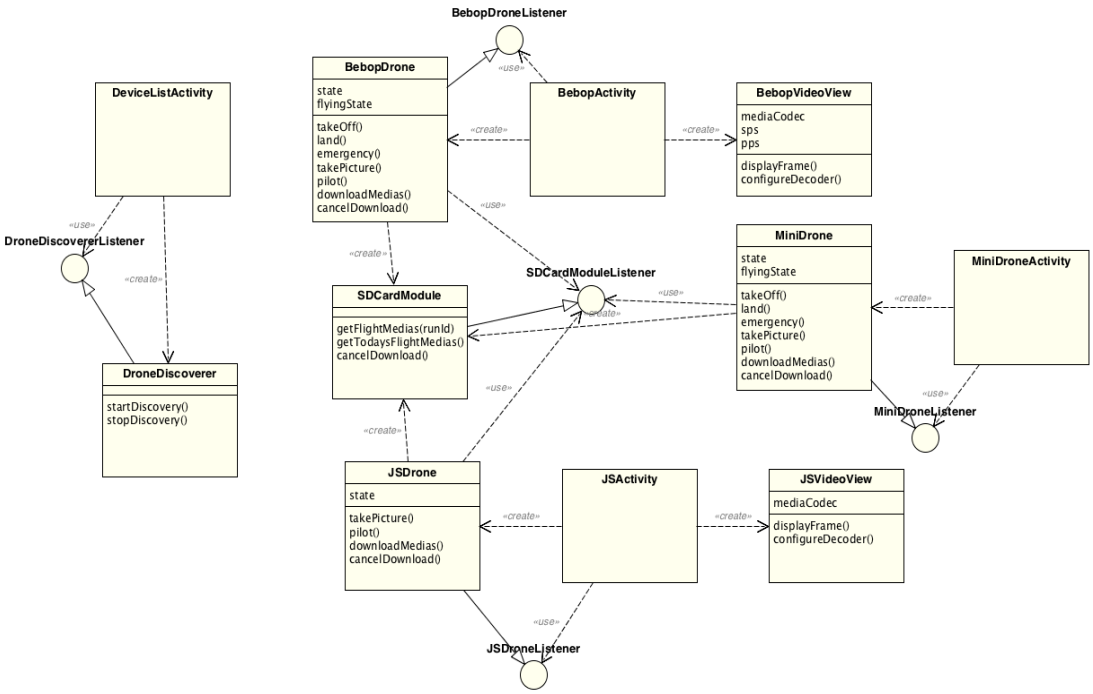
Nous l'avons utilisée afin de se familiariser avec la Jumping Sumo puis la Rolling Spider et pour mieux comprendre leurs fonctionnements.

Pour l'utiliser, le logiciel a besoin de reconnaître les drones, pour cela il suffit simplement de les allumer afin d'émettre un signal Wifi et Bluetooth. Si nous ne reconnaissons toujours pas les drones utilisés cela peut être un problème de Firmware, il suffira donc de les mettre à jour pour que l'application puisse les détecter sans problème.

Une fois les drones testés, nous avons décidé d'installer le SDK de Parrot pour obtenir des exemples de programme.

2) L'ARSDK3 :

Ci-dessous l'architecture de l'ARSDK3 :

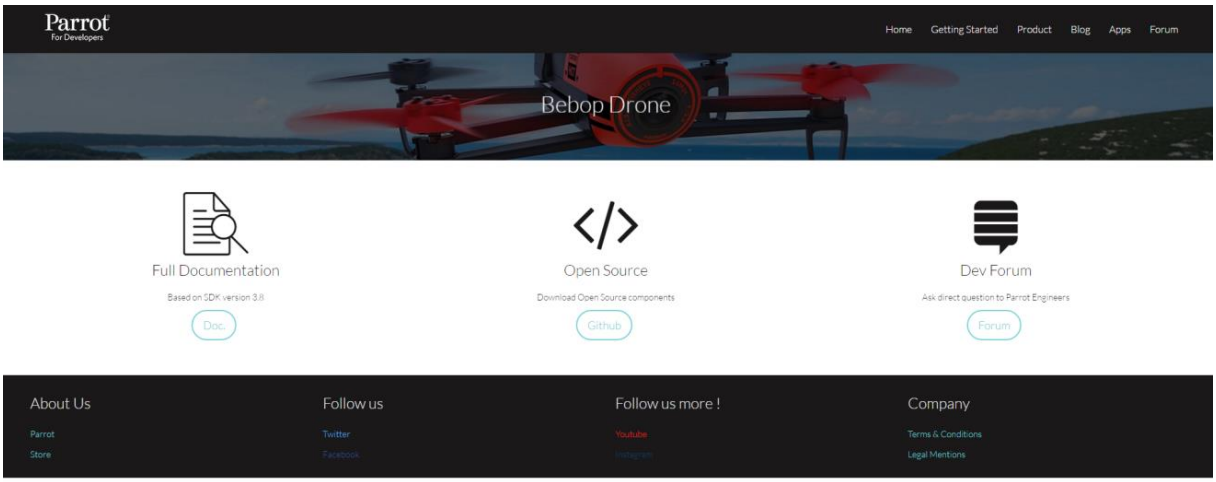


Pour notre projet, nous nous intéresserons donc à la partie JSDrone .Cet SDK nous permet d'avoir la possibilité d'utiliser plusieurs langages de programmation pour l'utilisation des drones tels que :

- Le langage Objective-C pour l'utiliser sur l'Iphone
- Le langage Java sur Android
- Le langage C sur Standalone

Pour l'installation du SDK, nous avons suivi le site des développeurs de Parrot ci-dessous:

"**http://developer.parrot.com/docs/bebop/?java#how-to-build-the-sdk**"



Ce lien nous permet de suivre les étapes importantes pour télécharger toutes les sources du SDK en fonction du système d'exploitation utilisé.

Pour cela, nous avons utilisé un système d'exploitation OSX Yosemite ainsi que Linux. Windows n'étant pas compatible.

Une fois toutes ces étapes effectuées, nous pouvons récupérer les programmes des drones concernés dans notre répertoire : **"/monrepertoire/out/Unix-base/staging/usr/bin"**. Nous pouvons dès à présent avoir un aperçu de la programmation qui sera à effectuer. Une fois les programmes compilés, les drones peuvent être contrôlés à l'aide du clavier.

Concernant la configuration, Nous avons mis un certain temps avant d'obtenir correctement les programmes sur notre machine, principalement avec le système d'exploitation Linux. En effet, certaines bibliothèques étaient manquantes et des erreurs survenaient régulièrement. Ensuite, une fois le programme compilé, celui-ci n'arrivait pas à obtenir le chemin de certaines librairies, il suffisait donc d'ajouter certains chemins pour que ceux-ci puissent reconnaître les librairies installées.

III) TECHNOLOGIES UTILISEES :

Le SDK étant assez compliqué, nous avons décidé de nous tourner sur d'autres méthodes plus accessibles pour l'élaboration de notre programme.

1) NodeJS :

Le nodeJs permet de développer des applications WebSocket (client/serveur) en javascript très facilement. Son avantage est la possibilité de s'exécuter en dehors d'une page web et grâce à sa gestion des événements, cela reste très efficace. De plus, l'optimisation apportée par Google lors du développement de la machine virtuel V8 augmente sa vitesse d'exécution.

Voici le code qui permet d'initialiser notre serveur et d'attacher une socket à notre serveur :

```
23  // Chargement du fichier index.html affiché au client
24  var server = http.createServer(function(req, res) {
25      fs.readFile('./test.html', 'utf-8', function(error, content) {
26          res.writeHead(200, {"Content-Type": "text/html"});
27          res.end(content);
28      });
29  });
30
31
32  // Chargement de socket.io
33  var io = require('socket.io').listen(server);
34
```

Afin d'écouter sur le port 8080 nous devons écrire:

```
240  server.listen(8080);
```

Voici comment sont gérés les événements en nodeJS :

```
114  io.sockets.on('connection', function (socket) {
115      console.log('Un client est connecté !');
116      socket.emit('message', 'Vous êtes bien connecté !');
117
118
119
120
121      socket.on('refreshServeur', function (message) {
122          console.log('Un client me parle ! Il me dit : ' + message);
123      });
```

Tout d'abord, nous avons besoin d'initialiser la socket avec "sockets.on" puis de créer à l'intérieur de cette fonction le traitement que l'on doit réaliser sur la donnée obtenu.

Si notre serveur reçoit 'refresh serveur' on affichera alors "un client me parle ! Il me dit :" et l'ensemble du message que le client à envoyer.

Du côté client, dans notre page web, il suffit d'initialiser notre socket à localhost:8080.

```
54        <script>
55              var socket = io.connect('http://localhost:8080');
56              var batteryLvl;
57              var onGround = true;
58              socket.on('message', function(message) {
59                    alert('Le serveur a un message pour vous : ' + message);
60              })
```

Afin de communiquer avec le serveur, le client peut envoyer et écrire des messages :

```
235             function connect()
236             {
237                   //on ne peut pas mettre 'connect' en parametre, le mot est déjà pris ... ou bug ...
238                   socket.emit('connectedRollingSpider', 'Connecte toi a la rolling spider');
239             }
```

Nous avons décidé d'utiliser le nodeJS puisqu'il a fallu trouver un moyen de centraliser toute notre gestion des drones afin que l'on puisse faire du traitement sur les informations transmises et réceptionnées.

De plus, des développeurs avaient déjà utilisé le nodeJS dans leurs applications Rolling Spider et Jumping Sumo puis nous avions gérer cela à l'aide d'une page internet.

2) Bluetooth Low Energy :

Le Bluetooth Low Energy a été crée en 2011 par “the Bluetooth Special Interest Group” (SIG).

Voici la comparaison entre le Bluetooth normal et le Bluetooth Low Energy :

Technical Specification	Classic Bluetooth technology	Bluetooth Smart technology
Distance/Range (theoretical max.)	100 m (330 ft)	>100 m (>330 ft)
Over the air data rate	1–3 Mbit/s	1 Mbit/s
Application throughput	0.7–2.1 Mbit/s	0.27 Mbit/s
Active slaves	7	Not defined; implementation dependent
Security	56/128-bit and application layer user defined	128-bit AES with Counter Mode CBC-MAC and application layer user defined
Robustness	Adaptive fast frequency hopping, FEC, fast ACK	Adaptive frequency hopping, Lazy Acknowledgement, 24-bit CRC, 32-bit Message Integrity Check
Latency (from a non-connected state)	Typically 100 ms	6 ms
Minimum total time to send data (det.battery life)	100 ms	3 ms [31]
Voice capable	Yes	No
Network topology	Scatternet	Scatternet
Power consumption	1 W as the reference	0.01 to 0.5 W (depending on use case)
Peak current consumption	<30 mA	<15 mA
Service discovery	Yes	Yes
Profile concept	Yes	Yes
Primary use cases	Mobile phones, gaming, headsets, stereo audio streaming, smart homes, wearables, automotive, PCs, security, proximity, healthcare, sports & fitness, etc.	Mobile phones, gaming, smart homes, wearables, automotive, PCs, security, proximity, healthcare, sports & fitness, Industrial, etc.

On remarque que le BLE a été développé avant tout pour transmettre des petites informations avec une consommation basse, ce qui impact le temps total pour transmettre des données.

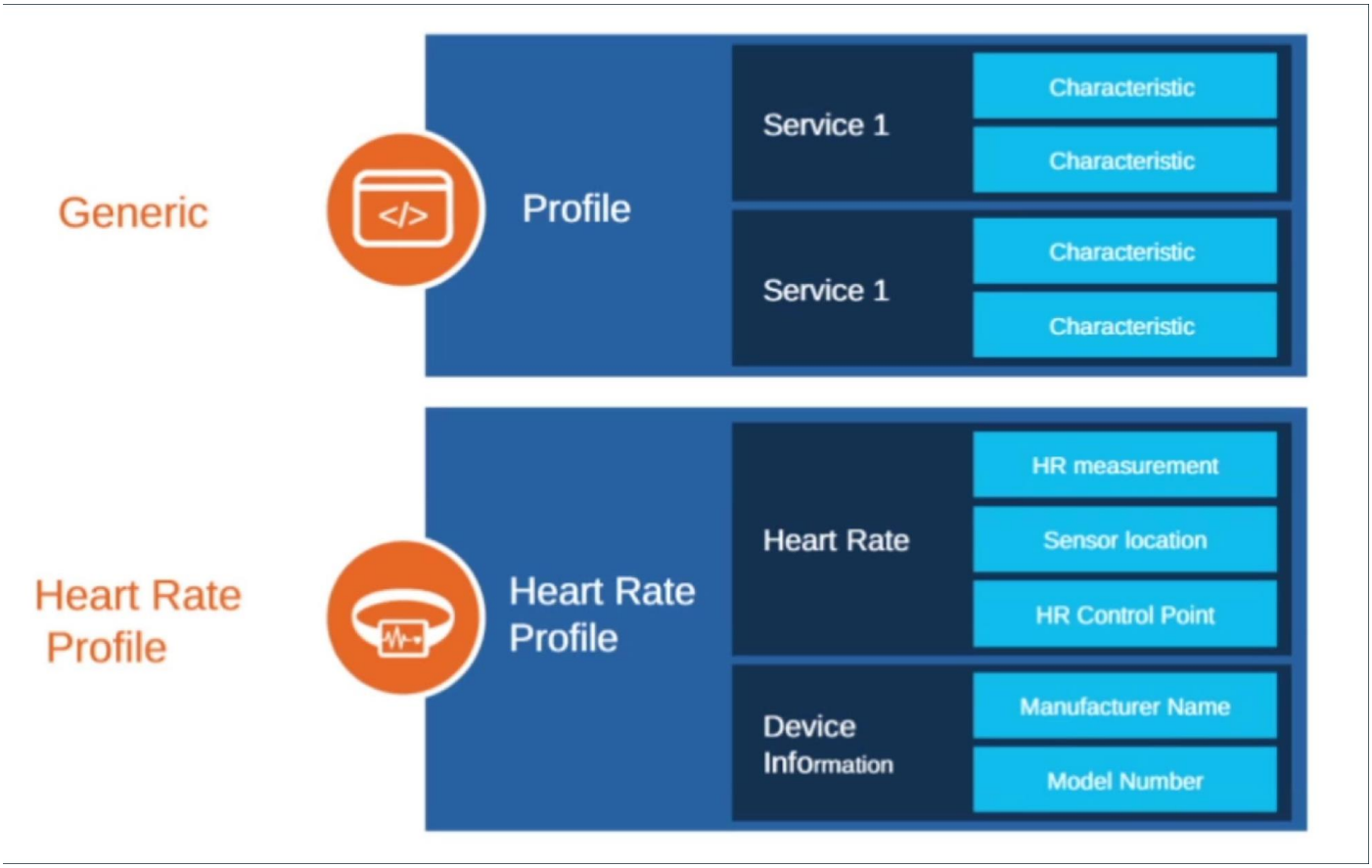
Son fonctionnement :

Le concept de BLE repose essentiellement sur la création d’un “gestionnaire” des informations émises et réceptionnées.

Cela s’appelle le GATT (The Generic Attribute Profile), il va de plus gérer la communication entre plusieurs appareils. Il est composé de profiles, de services, de caractéristiques et de descripteurs.

On va pouvoir créer un profil qui correspond à notre application. Ce profil comprendra des services qui correspondront à nos capteurs, par exemple celui d’un cardio-mètre, puis de créer des caractéristiques contenant les informations relatives au capteur que l’on récupère avec le “descriptor”.

Voici un schéma afin d'illustrer la conception d'une application Bluetooth Low Energy :



Pour cela nous utilisons Noble, un module de nodeJS qui sert à scanner le réseau Bluetooth avec la possibilité de communiquer avec les appareils détectés *(voir l'annexe pour un morceau de code avec la possibilité d'afficher toutes les informations de la Rolling Spider qui nous a permis de savoir comment la société Parrot a conçu cette communication).*

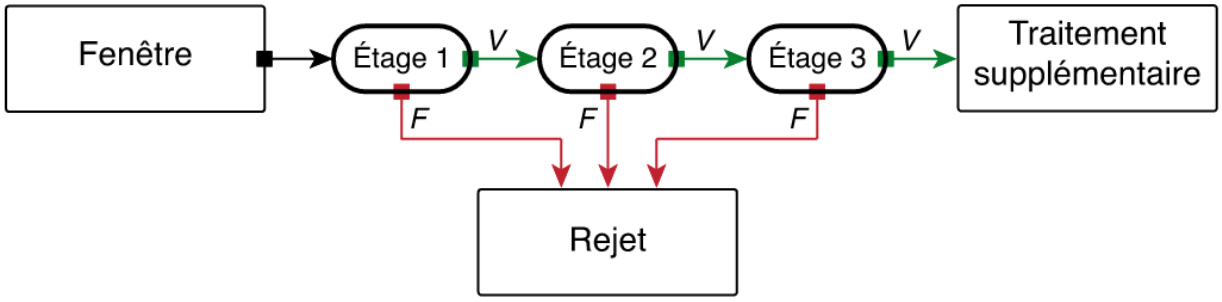
Ensuite, la caméra de la Rolling Spider est difficilement accessible contrairement à la JumpingSumo. Nous avons tout d'abord détecté le drone volant à l'aide de la caméra du drone roulant. Pour cela, nous nous sommes exercés en utilisant la bibliothèque Open CV pour une détection faciale.

3) Open CV :

L'open CV est une bibliothèque graphique libre spécialisée dans le traitement d'image en temps réel. Nous l'utiliserons ici pour traiter les images provenant de la caméra du drone pour concevoir une détection d'objet.

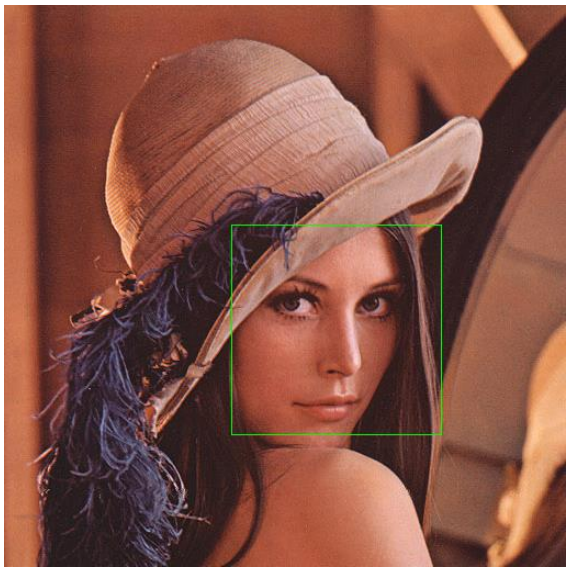
Pour cela, nous pouvons utiliser la méthode de Viola et Jones qui est un algorithme de détection d'objet en temps réel.

Nous utiliserons ici l'algorithme de Viola et Jones pour la détection d'objet. Cette méthode a besoin de quelques centaines d'exemples de l'objet que l'on veut détecter par apprentissage supervisé, c'est-à-dire que l'on cherche à produire des règles à partir d'une base de données contenant des exemples automatiquement. Les exemples entrainent un classifieur pour détecter l'objet en testant la présence à toutes les positions et à toutes les échelles. Comme on peut voir ci-dessous, les « Fenêtres » sont traitées séquentiellement par les classifieurs, s'il n'y a pas de similitude, la fenêtre est rejetée immédiatement.



La méthode parcours donc l'ensemble de l'image. Plus il y a d'échantillon, plus nous avons de chance de reconnaître l'objet.

Ci-dessous un exemple de détection d'image :



Pour cela, nous pouvons récupérer la distance à l'aide du carré affiché sur nos images récupérées. En effet, nous avons remarqué que pour une distance de 70cm, X=82 et Y=82 et pour une distance de 140cm nous avons X=40 et Y=40. Il suffit donc de faire le calcul suivant :

Distance=((XYref*Distanceref)/XY)*coefficient

Avec:

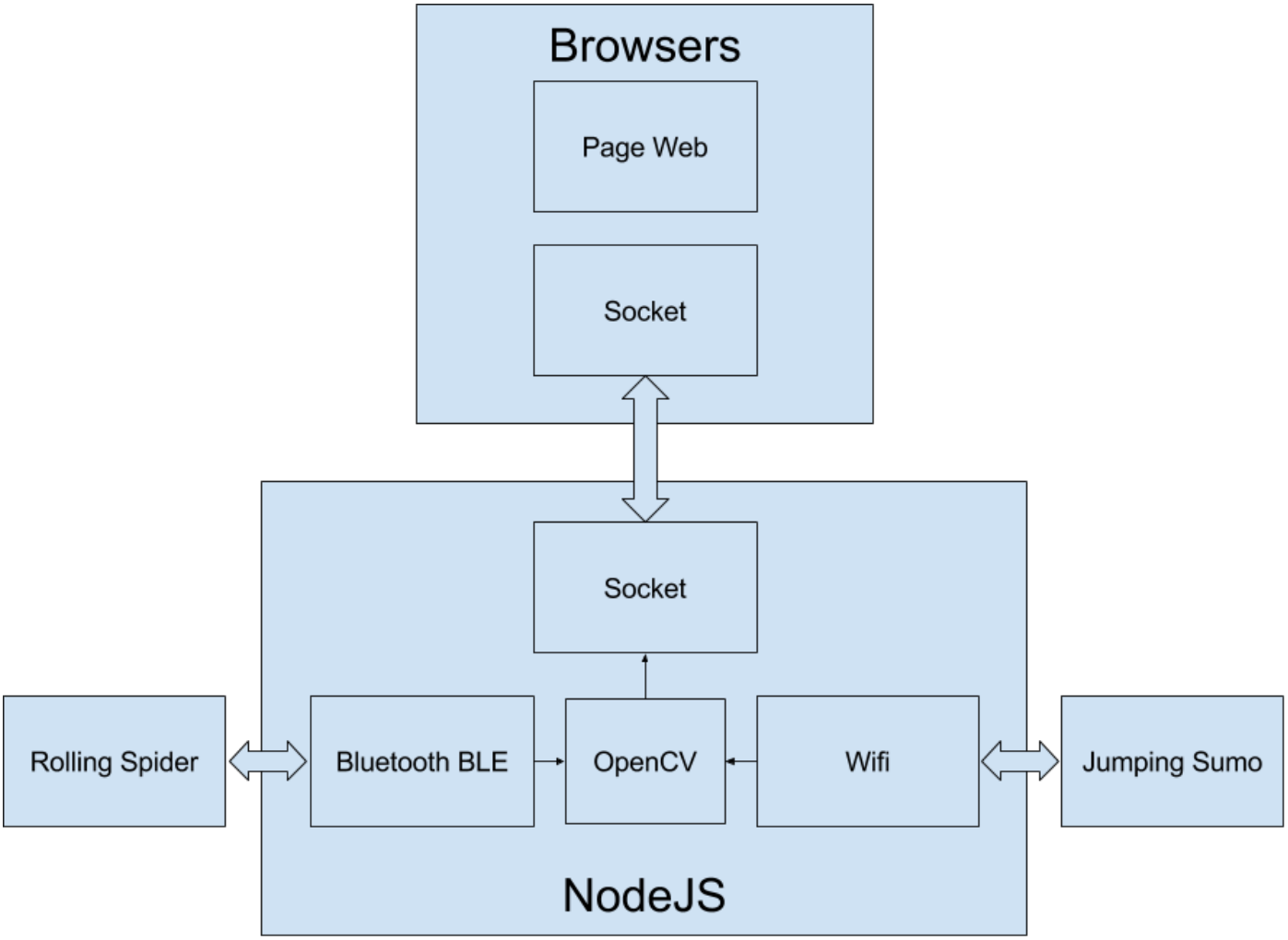
- Distanceref=70
- XYref=82
- Distance= la distance de la cible en temps réel
- XY=la taille du carré en temps réel
- coefficient= le coefficient choisi pour obtenir la distance de la cible en temps réel (ici nous prendrons 0.81) .

Dans un premier temps, notre objectif est le suivi d'un visage en fonction de la distance à l'aide de la Jumping Sumo. Le drone avance si la distance est supérieure à 70cm et recule si celle-ci est inférieure à 30cm.

Ensuite le drone effectue une rotation si le visage se déplace à gauche ou bien à droite.

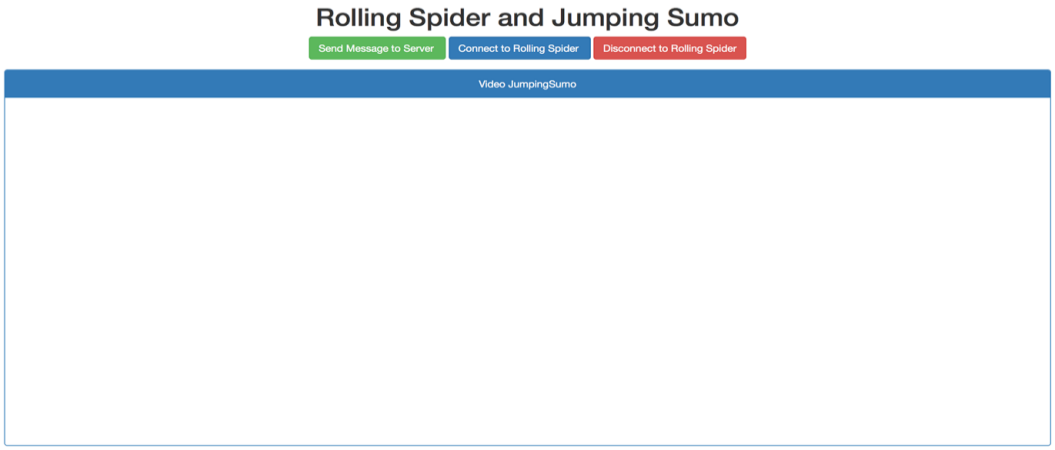
IV) APPLICATIONS :

Voici un schéma fonctionnel d'une application des drones.



Le contrôle des drones:

Nous avons tout d'abord élaboré une première application pour un contrôle des deux drones en même temps. Ci-dessous l'interface graphique :



L'interface comprend :

- Un bouton de connexion au serveur
- Un bouton de connexion de la Rolling Spider
- Un bouton de déconnexion de la Rolling Spider
- Un emplacement pour la vidéo de la Jumping Sumo

Voici les boutons du clavier nécessaires pour le contrôle des drones :

En bleu nous contrôlerons la **Rolling Spider** et en rouge **la Jumping Sumo.**

Z ➔ Avancer

S ➔ Reculer

Q ➔Aller à gauche

D ➔ Aller à droite

Espace ➔ décollage

P ➔ Voler vers le haut

M ➔Voler vers le bas

I ➔ Avancer

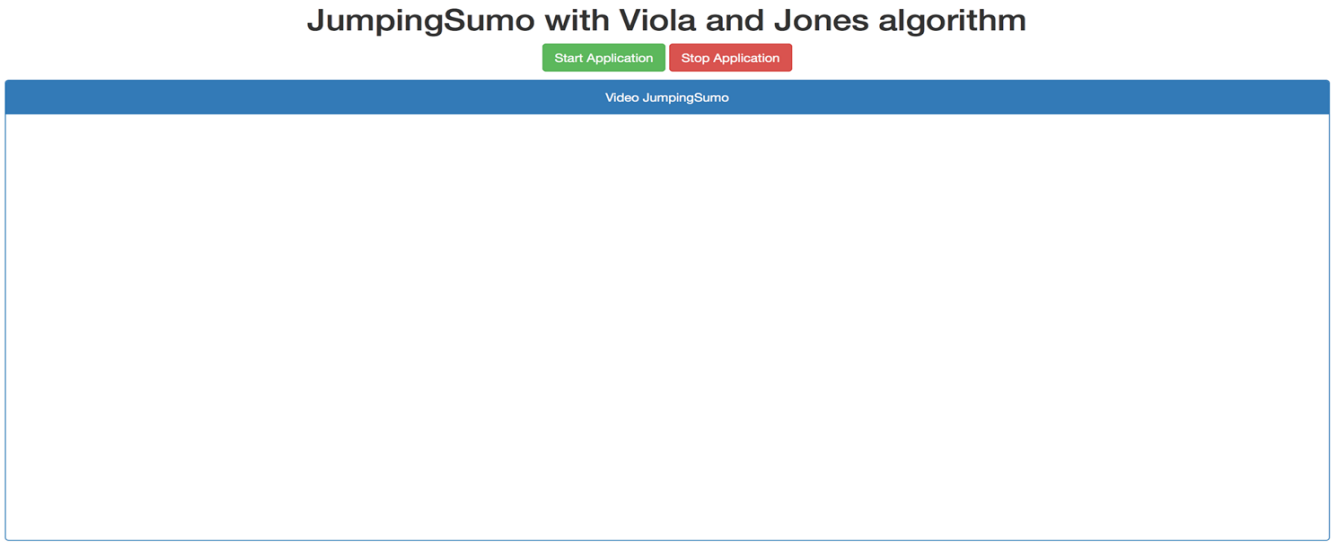
K ➔ Reculer

J ➔ Aller à gauche

L ➔ Aller à droite

Le tracking avec la Jumping Sumo :

Nous avons ensuite élaboré une deuxième application pour repérer la détection provenant de la caméra de la Jumping Sumo. Ci-dessous l'interface graphique :



La vidéo de droite concerne la vidéo prise en temps réel, celle de gauche est celle avec la détection d'objet. S'il n'y a pas de détection, la vidéo reste inactive.

Pour conclure le projet de notre deuxième année de cycle ingénieur en Informatique Microélectronique et Automatique, celui-ci fut très inspirant et intéressant.

Malheureusement, nous n'avons pas réussi à atteindre notre objectif final qui est bel et bien le drone volant suivant le drone roulant, nous avons consacré du temps à essayer de trouver une solution afin de récupérer la caméra du drone volant mais cela reste très compliqué voire très peu faisable.

Le moyen de suivi, qu'on aurait pu améliorer, est d'utiliser cette méthode de détection en ajoutant plusieurs images de la Rolling Spider pour que le drone roulant puisse suivre le drone volant. Mais cela reste assez limité au niveau du champ de vision de la jumping sumo.

Néanmoins, ce fut une très belle expérience et un réel plaisir d'en apprendre davantage sur la programmation d'un drone de la société Parrot et sur le tracking d'une image.

V) ANNEXE:

bluetoothTest.js

```
var noble = require('noble');


var serviceUUIDs;
var characteristicUUIDs;

noble.on('stateChange', function(state) {
  if (state === 'poweredOn') {
    noble.startScanning();
  } else {
    noble.stopScanning();
  }
});

noble.on('discover', function(peripheral) {


  console.log('peripheral: ' + peripheral.advertisement.localName);

  var macAddress = peripheral.uuid;
    var rss = peripheral.rssi;
    console.log('found device: macAddress ' + macAddress + ' rss   '+ rss + 'is rolling spider ' +isRollingSpider(peripheral));

    console.log('Found device with local name: ' + peripheral.advertisement.localName);



  peripheral.connect(function(error) {
      console.log('connected to peripheral: ' + peripheral.uuid);
      console.log('peripheral state : ' + peripheral.advertisement.state);

      peripheral.discoverServices(null, function(error, services) {
        console.log('discovered the following services:');
        for (var i in services) {

          console.log('  ' + i + ' services: ' + services[i]);
          console.log('  ' + i + ' uuid: ' + services[i].uuid);
          services[i].once('includedServicesDiscover', function(includedServiceUuids)
          {
            console.log(' includedServiceUuids  ' + includedServiceUuids);
          });
          services[i].discoverCharacteristics([], function(err, characteristics) {

          for(var i in characteristics)
          {
            console.log('found characteristic:'+ characteristics[i]);
          }
        });

    }

    });
  });
});
```

Code pour l'affichage de toutes les informations de la Rolling Spider