**AR DRONE – NODE JS**

Nom du dossier : Projet en cours

Nom des dossiers internes :

- drone //contient tout ce qui concerne le drone

- node\_modules //contient les différents modules node js

- public //contient tout ce qui concerne le server (la page web)

Nom des fichiers internes :

- client.js //contient tout ce qui concerne le client (le drone)

- Server.js //contient tout ce qui concerne le server (la page web en html)

On va seulement utiliser server.js, client.js et index.html (./public/index.html).

**Server.js :**

//Utilisation de la bibliothèque express qui gère tout ce qui est framework (= bibliothèque)

var express = require('express')

var app = express()

//Creation d'un server de type http pour app

var server = require("http").createServer(app)

//Indique que le dossier public contient des fichiers statiques : images

app.use(express.static(\_\_dirname + '/public'));

//Require 2 fichiers : camera-feed et controller

require("./drone/camera-feed");

require("./drone/controller");

//Ecoute sur le port 3000

app.listen(3000);

La principale fonction de server.js est de créer un server où on pourra interagir avec le drone : les vidéos, l’altitude, le niveau de batterie, contrôle du drone avec boutons et flèches.

**Client.js :**

//Utilisation de la bibliothèque ar drone

var arDrone = require('ar-drone');

//Utilisation de la bibliothèque socket.io qui s'occupe de tout ce qui est temps réel

var io = require('socket.io-client'),

//On se connecte au server local 3111

socket = io.connect('localhost', { port: 3111});

//On affiche dans la console la connection

socket.on('connect', function () { console.log("socket connected"); });

//Socket emet une image de nom : drone

socket.emit('image', { image: 'drone' });

//On créé un client de notre drone

var client = arDrone.createClient();

//On demande au client d'enregistrer des images de type PNG

client.getPngStream()

//En cas d'erreur, l'afficher dans la console

.on('error', console.log)

//En cas de donnée, socket emettra sur le server les images de notre camera

.on('data', function(frame) {

socket.emit('image', { image: frame });

});

**Index.html:**

Script:

$(function () {

function startArDRoneStream()

{

new NodecopterStream(document.getElementById("placeholder"), {port: 3001});

}

function startArDroneController()

{

var socket = io.connect('http://localhost:3002');

socket.on('connect', function () { // TIP: you can avoid listening on `connect` and listen on events directly too!

console.log("Connection Successful");

});

socket.on('event', function (data) {

if(data.name=="battery"){

$("#battery-indicator").css('width',data.value+'%');

$("#battery-value").html(data.value+'%');

}

});

/\*socket.on('event', function (data) {

if(data.name=="altitude"){

$("#altitude-indicator").css('width',data.value+'m');

$("#altitude-value").html(data.value+'m');}

});\*/

$("#takeoff").click(function(){

console.log("Asking Server to send takeoff command to Ar Drone");

socket.emit('event',{name:"takeoff"});

});

$("#spin").click(function(){

console.log("Asking Server to send spin command to Ar Drone");

socket.emit('event',{name:"spin"});

});

$("#stop").click(function(){

console.log("Asking Server to send stop command to Ar Drone");

socket.emit('event',{name:"stop"});

});

$("#land").click(function(){

console.log("Asking Server to send land command to Ar Drone");

socket.emit('event',{name:"land"});

});

}

startArDRoneStream();

startArDroneController();

Pour sélectionner la caméra du dessous, il faut rajouter une ligne de code. Cela va configurer notre drone :

Il faut aller dans la bibliothèque « Ar-Drone », puis ouvrir « index.js » et rajouter la ligne :

//Bottom camera

client.config('video:video\_channel', 3);

Il faut la rajouter dans la fonction qui s’occupe du client (c’est-à-dire notre drone) :

exports.createClient = function(options) {

var client = new arDrone.Client(options);

//Select bottom camera

client.config('video:video\_channel', 3);

client.resume();

return client;

};