UNIVERSITA' DEGLI STUDI DI
PERUGIA
DIPARTIMENTO DI MATEMATICA E
INFORMATICA

Laurea Triennale in Informatica

Uso dell’intelligenza artificiale per il controllo remoto del drone Parrot ardrone 2.0.

Artificial intelligence use for remote control of drone Parrot ardrone 2.0.

Autore:
Antonio Malena

Relatore:
Prof. Stefano Bistarelli

Anno Accademico 2016-2017

Ringraziamenti

La tesi è dedicata a tutte le persone che mi hanno sostenuto e aiutato durante questi anni di università: famiglia, amici, colleghi e professori. Ringrazio il relatore professore Stefano Bistarelli, che ha ritenuto possibile la realizzazione della mia proposta di lavorare sul drone per il tirocinio e successivamente per la tesi, aiutandomi con idee per lo sviluppo e spronandomi ad insistere sulla risoluzione dei problemi che si sono presentati. Ringrazio gli amministratori di rete e in particolar modo i tecnici del laboratorio del dipartimento: fabio e riccardo, senza i quali la conclusione di questo progetto non sarebbe stata possibile e nonostante i loro impegni, mi hanno permesso di utilizzare i loro computer concedendomi la loro esperienza e il tempo per riuscire a superare i problemi legati alla rete, consentendomi così di accrescere la mia esperienza. Un grazie va ai miei genitori e la mia famiglia che mi sono stati sempre vicini, nonostante le distanze, sostenendomi economicamente, ma soprattutto moralmente insistendo e spronandomi a studiare costantemente, nonostante la mia ostinazione nel non seguire mai i consigli. Un ringraziamento va ai colleghi che hanno condiviso con me questo percorso, con i quali ho condiviso le gioie e le delusioni di ogni esame superato. Infine ringrazio tutti i miei amici vicini e lontani.

INTRODUZIONE

Obiettivo di questo progetto è stato studiare il modo di inviare comandi al drone parrot ardrone 2.0 attraverso una pagina web con lo scopo di usare l'intelligenza artificiale per riconoscere e seguire un colore autonomamente. Concepito per essere pilotato dagli smartphone sfruttando la sua rete hotspot, si è intervenuti sul modulo wifi per modificare l'architettura di rete, al contrario di un'architettura punto a punto, è stato effettuata una connessione con la rete wi-fi (di unipg), in modo da poter pilotare il drone da remoto da un qualsiasi dispositivo connesso alla rete, dell'ateneo in questo caso, ma in realtà in una qualunque rete connessa ad internet, estendendo così la portata del segnale. Ultimo passo del progetto è stato fare in modo che il drone riconoscesse un colore tramite la sua videocamera e lo seguisse in volo autonomamente, quindi analizzando l'immagine dal browser, in base alla posizione di un oggetto colorato, si inviano i comandi da eseguire.

CAPITOLO 1
IL DRONE: PARROT ARDRONE 2.0

Figura 1 - ardrone 2.0

1.1 Presentazione

Realizzato da Parrot SA[1], il drone ardrone2.0 (Figura 1) è l'ultima versione della serie "ardrone", dotato dello stesso design della versione precedente, in questa vengono apportati notevoli miglioramenti tra i quali la telecamera in grado di filmare in 720p. Il corpo è realizzato con un particolare tipo di polimero, denominato PPE (Polifeniletere), è molto leggero ed altamente resistente; dotato di 4 motori brushless da 15 watt (Figura 3) riesce ad avere una buona stabilità in volo grazie ai sensori, gestiti da un computer di bordo sul quale gira Linux 2.6.32, al quale si accede in wifi tramite telnet. Sviluppato per essere usato da smartphone ios/android è gestito/pilotato tramite l'app proprietaria AR.FreeFlight sfruttando una rete hotspot alla quale è possibile collegarsi da molti dispositivi.

Figura 2 - batteria agli ioni di litio

L'alimentazione è fornita da una batteria agli ioni di litio da 1000mAh (Figura 2), che purtroppo consente solo 10 minuti circa di autonomia, ma può essere sostituita da una più potente non presente nella dotazione standard. Infine presenta due scocche, una per il volo outdoor, l'altra per il volo indoor che attenua l' evetuale impatto delle eliche.

Di seguito si riportano le figure[2] della componentistica interna del drone:

1 Figure da https://www.parrot.com
2 Figure da http://www.ratsaas.com/quadrotor

La scheda madre (Figura 4) sulla quale sono montati il processore arm cortex con la ram da 1GB, il connettore usb, il modulo wifi e gli ingressi per i connettori dei motori e delle telecamere, ed un connettore per la daughterboard. La daughterboard (Figura 5) presenta numerosi sensori, tra i quali: sensore di pressione, accellerometro, giroscopio e un microcontrollore; infine è presente un connettore per il sensore di ultrasuoni (Figura 6), modulo visibile nella parte inferiore del drone, dotato di un componente trasmettitore e l'altro ricevitore, utile per stabilire l'altezza da terra.

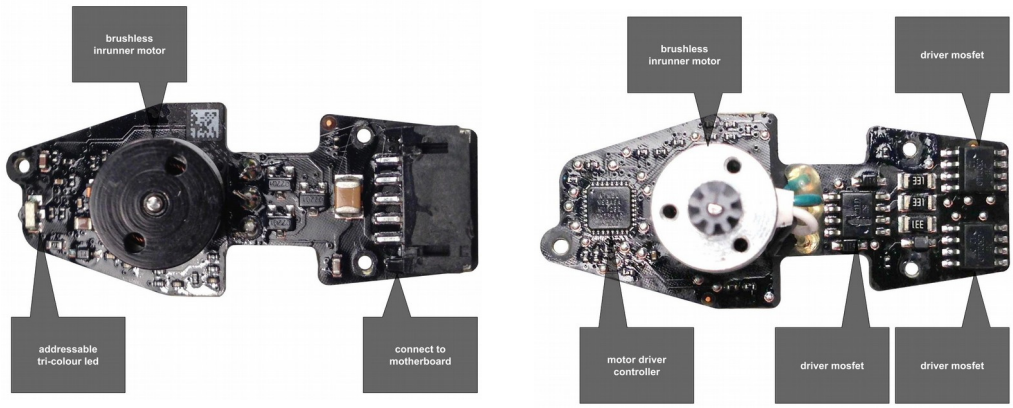

Figura 3 – motori brushless da 15 watt

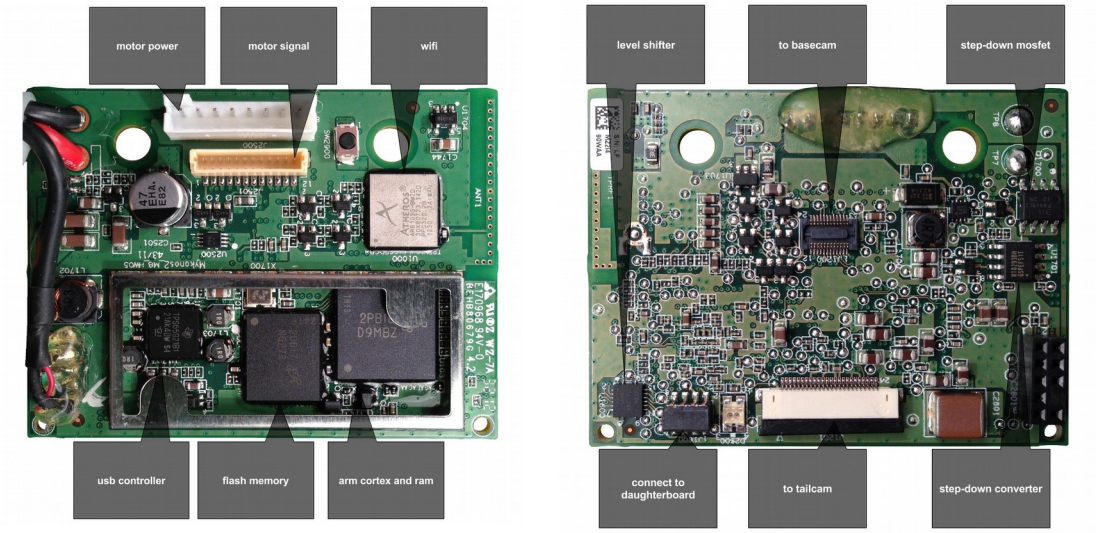

Figura 4 – scheda madre

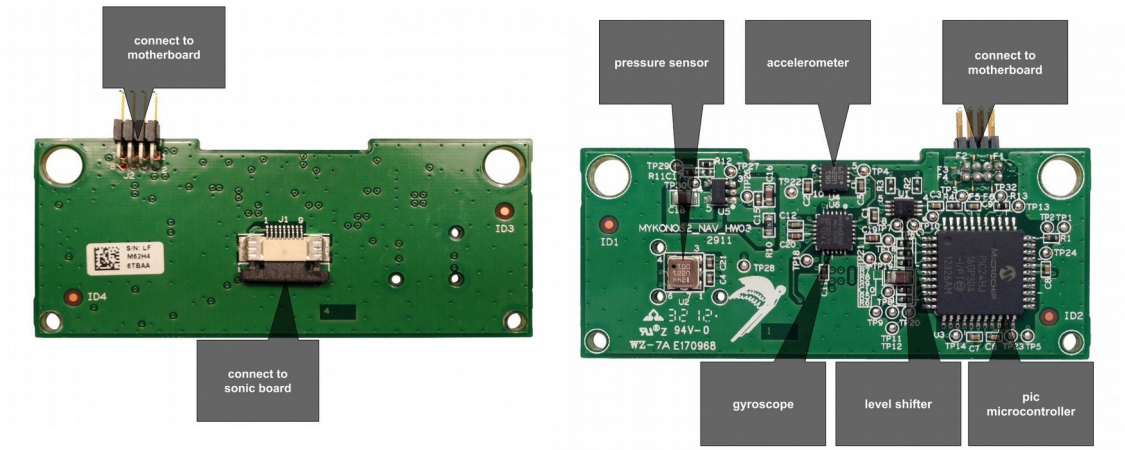

Figura 5 - daughterboard

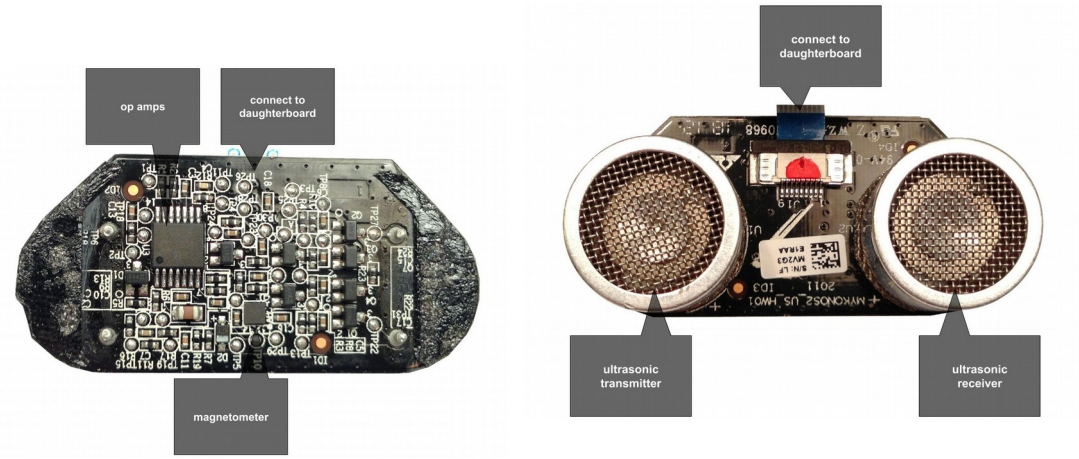

Figura 6 – Ultrasonic sensor

1.2 Volo e movimento

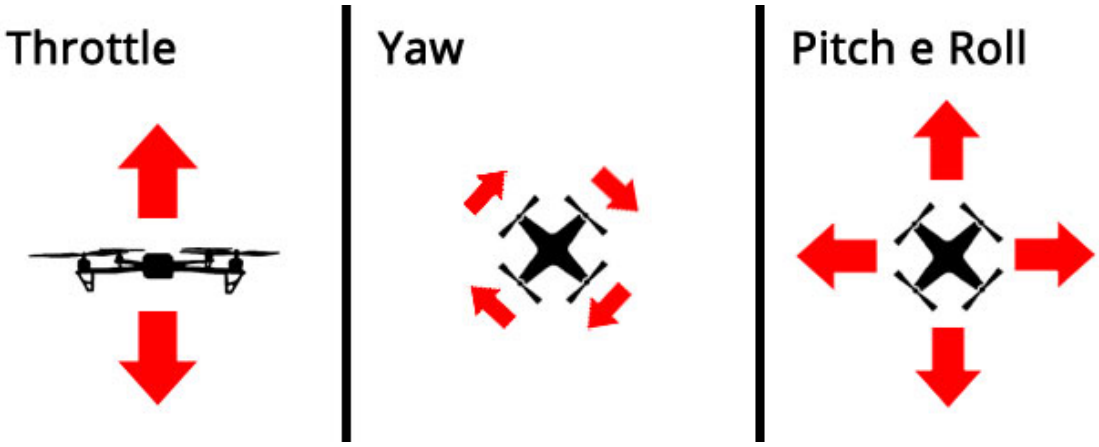

Figura 7 – I movimenti sugli assi[3]

Il movimento in volo del quadrirotore e quindi il modificarsi della posizione sugli assi (Figura 7) è consentito dalla velocità sincronizzata dei quattro motori e quindi delle eliche che determinano PITCH (beccheggio), ROLL(rollio), YAW (imbardata), THROTTLE.

I motori del quadrirotore girano in senso antiorario e orario a due a due (Figura 8), come si può notare in figura, questo permette il contrasto alla forza centrifuga dei motori conferendo allo stesso stabilità. Quando la velocità di rotazione delle eliche permette di avere una portanza capace di contrastare il peso del quadricottero, esso inizierà a sollevarsi da terra, l'aumento/diminuzione della velocità sincronizzata dei 4 motori permette il movimento sull'asse verticale: **THROTTLE**.

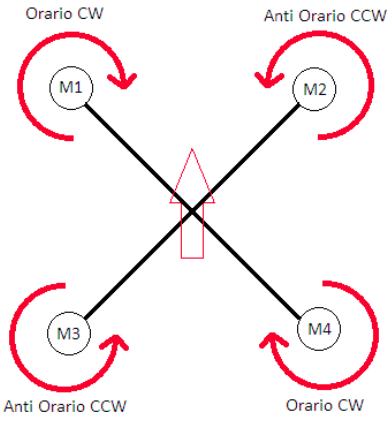

Figura 8 – rotazione motori

3 Figure da http://www.projectems.it/esercizi-per-imparare-a-pilotare-i-droni/

PITCH è lo spostamento in avanti/dietro del drone, consentito dalla riduzione della velocità dei due motori anteriori e l'aumento dei due posteriori o viceversa, visivamente il drone si inclina per procedere nella direzione voluta. Stessa cosa vale per il ROLL che vede però coinvolta la sincronizzazione dei motori di destra e sinistra. Infine con YAW si intende una rotazione sull'asse verticale, consentita dalla sincronizzazione incrociata delle velocità dei motori.

1.3 Connessione Wi-Fi

A differenza della maggiorparte dei droni in commercio che ricevono i comandi tramite segnali radio, l'ardrone si serve del modulo wifi emettedo una rete hotspot alla quale è possibile connettersi da un qualunque dispositivo, ottenendo così una connessione di tipo punto a punto. Quindi si ottegono piccole reti locali smartphone-drone o pc-drone con una portata massima equivalente alla portata del segnale Wi-Fi, inferiore di molto alla portata del segnale radio.

1.4 Applicazioni

Il drone ARDrone è una via di mezzo fra un elicottero da modellismo ed un semplice giocattolo. Presenta la possibilità di essere collegato a un dispositivo Apple o android e avere uno steaming delle immagini che possono essere 'virtualizzate' ed usate per creare giochi come combattimenti o gare in percorsi. L'interfaccia grafica del software di controllo AR.FreeFlight (app per smartphone), mostra all'utente le immagini riprese dalla videocamera, due joystick per il pilotaggio (o in alternativa un joystick ed un pulsante per attivare l'uso degli accelerometri del dispositivo di comando), l'informazione sulla carica rimanente della batteria, un pulsante per l'atterraggio/decollo, un pulsante per

l’interruzione dei motori in caso di emergenza e due icone: una per l’accesso nel menù di personalizzazione e una per l’aggiornamento/cambio di visuale della videocamera. Per la programmazione, invece esiste la possibilità di scaricare l’ sdk in C con un dettagliato manuale, ma sono presenti online moduli più semplici da utilizzare.

CAPITOLO 2

IL LINGUAGGIO E LE LIBRERIE (MODULI)

2.1 Nodejs

Node.js[4] è un framework per realizzare applicazioni Web in JavaScript, permettendoci di utilizzare questo linguaggio, tipicamente utilizzato nella "client-side", anche per la scrittura di applicazioni "server-side". La piattaforma è basata sul JavaScript Engine V8, che è il runtime di Google utilizzato anche da Chrome e disponibile sulle principali piattaforme, anche se maggiormente performante su sistemi operativi UNIX-like. La caratteristica principale di Node.js risiede nella possibilità che offre di accedere alle risorse del sistema operativo in modalità event-driven e non sfruttando il classico modello basato su processi o thread concorrenti, utilizzato dai classici web server.

Il modello event-driven, o "programmazione ad eventi", si basa su un concetto piuttosto semplice: si lancia una azione quando accade qualcosa. Ogni azione quindi risulta asincrona a differenza dei pattern di programmazione più comune in cui una azione succede ad un'altra solo dopo che essa è stata completata.

Ciò dovrebbe garantire una certa efficienza delle applicazioni grazie ad un sistema di callback gestito a basso livello dal runtime.

4 https://nodejs.org

2.2 Moduli

Node.js aderisce al più ampio progetto denominato CommonJS che si pone come obbietivo quello di avere un insieme di API coerenti e soprattutto standard per i molti interpreti JavaScript attualmente disponibili.

Le Specifiche CommonJS sono molte e di diversa natura, approvate o ancora in fase di progettazione, ovviamente oltre a CommonJS, il framework presenta una serie di API proprietarie che permettono di effettuare praticamente qualsiasi operazione a livello di rete e di socket, oltre alle librerie di alto livello come HTTP, SSL o DNS.

Importantissima risorsa dunque si rivela npm[5], repository di librerie scritte apposta per poter essere utilizzate con Node.js, facile da installare, presenta comandi per scaricare, ricercare ed aggiornare pacchetti software da utilizzare nelle proprie applicazioni.

2.3 node-ar-drone

Uno dei moduli essenziali per la realizzazione di questo progetto è node ar-drone[6], implementato da Felix Geisendörfer[7], può essere scaricato tramite npm, il progetto ancora in fase di sviluppo, dispone di ciò che è necessario per il controllo remoto del drone. Fase iniziale, utile all'apprendimento e consigliata dalla documentazione della libreria, è creare una REPL[8] (node shell), terminale che ci permette di eseguire comandi di base manualmente uno per volta. Quindi una volta installata

5 https://www.npmjs.com/
6 https://github.com/felixge/node-ar-drone
7 "http://felixge.de/"
8 https://nodejs.org/api/repl.html#repl_repl

la libreria, si crea un file javascript che chiameremo repl.js (Figura 9) contenente tre righe di codice,

```
var arDrone = require('ar-drone');
var client  = arDrone.createClient();
client.createRepl();
```

Figura 9 – repl.js

la prima serve a nodejs per trovare ricorsivamente la libreria ‘ar-drone’, la seconda inizializza il drone con tutti i collegamenti udp/tcp per il controllo dei comandi e la gestione delle componenti, la terza apre un terminale sul quale è possibile scrivere ed eseguire le istruzioni da inviare al drone (Figura 10). Da questi elementi base si passa al primo programma in grado di eseguire una lista di comandi autonomamente (Figura 11).

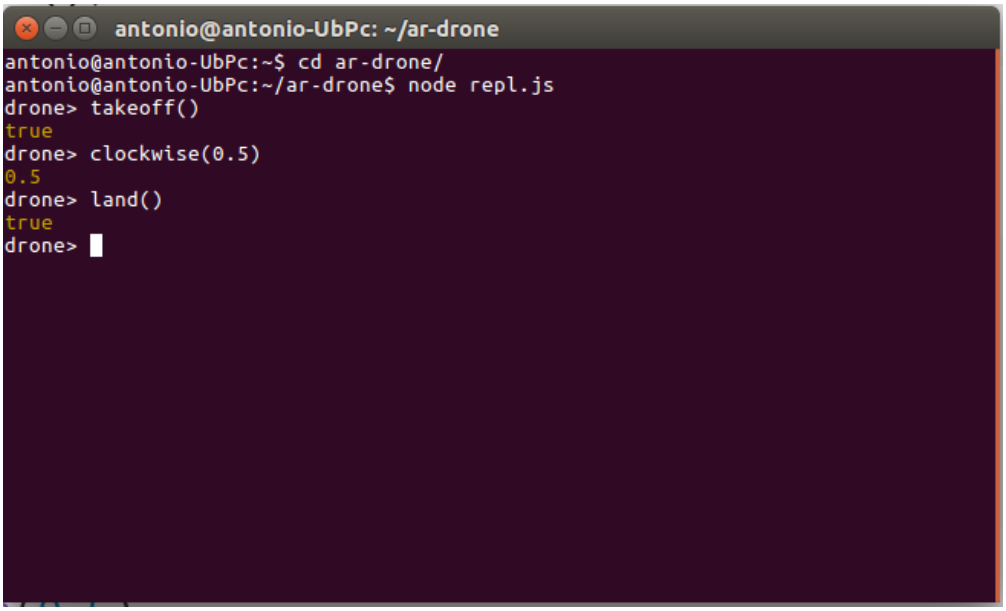

Figura 10 – Output repl.js

```
var arDrone = require('ar-drone');
var client  = arDrone.createClient();

client.takeoff();

client
  .after(5000, function() {
    this.clockwise(0.5);
  })
  .after(3000, function() {
    this.stop();
    this.land();
  });
```

Figura 11 – autonomusDrone.js

Dopo 5 secondi dal comado di decollo ”client.takeoff();” abbiamo uno YAW orario alla velocità di 0.5 “this.clockwise(0.5)”, dopo tre secondi abbiamo il comando di stop seguito dal comando di atterraggio. Qui di seguito si riporta una lista dei comandi principali:

-arDrone.createClient([options])

crea il client per inizializzare i comandi e le connessioni del drone.

-client.createREPL()

lancia un terminale per inviare i comandi al drone manualmente.

-client.getPngStream()

si utilizza per effettuare uno stream di immagini png catturati dalla camera del drone.

-client.getVideoStream()

effettua il collegamento per la trasmissione del video tramite tcp.

-client.takeoff(callback)

comando per il decollo, in assenza di altre istruzioni il drone si metterà in una situazione di stay and hover.

-client.land(callback)

comando per l’atterraggio

-client.up(speed) / client.down(speed)

comado per il THROTTLE, incremento/decremento dell’altitudine. La velocità è un numero compreso tra 0 e 1.

-client.clockwise(speed) / client.counterClockwise(speed)

comando per YAW, rotazione oraria e antioraria del drone su se stesso.

-client.front(speed) / client.back(speed)

comando per PITCH, movimento sul piano orizzontale, avanti/indietro.

-client.left(speed) / client.right(speed)

comando per ROLL,movimento sul piano orizzontale, destra/sinistra.

-client.stop()

comando per riportare il drone in una situazione di stay and hover.

2.4 dronestream

Nonostante la libreria sopra descritta abbia le funzioni riguardanti lo streaming di immagini o video si è notato che nell'utilizzarle si presenta un ritardo di 4 secondi nella trasmissione delle immagini tramite socket nel browser. Per ovviare questo problema che non permette il controllo remoto del drone, si ricorre al modulo dronestream[9] che garatisce un ottimo flusso di immagini senza ritardi, sul quale sarà possibile lavorare successivamente. L'ardrone trasmette il video su 192.168.1.1 (o qualunque indirizzo IP si configurerà) alla porta 5555, in un formato video h264 baseline[10] che viene inviato al browser tramite socket. Il browser non può leggere questo tipo di formato, perciò la libreria si serve del modulo Brodway.js[11] che si occupa del rendering su una WebGL canvas. Il modulo Broadway è stato realizzato a partire da un decoder h.264 esistente (nello specifico il decoder open source H.264 che Google utilizza in Android, scritto in linguaggio C), il quale è stato opportunamente semplificato e quindi compilato utilizzando Emscripten, uno strumento per compilare i bytecode LLVM[12] in JavaScript, questo decoder è in grado di visualizzare un video alla velocità di 30 frame al secondo, su hardware convenzionali ed eseguendo puramente codice javascript.

9 https://github.com/bkw/node-dronestream
10 https://it.wikipedia.org/wiki/H.264
11 https://github.com/mbebenita/Broadway
12 http://llvm.org/

2.5 Socket.io e Express

Infine si presentano i moduli che permettono la gestione e la connessione tra server e client: Express[13] è un framework che permette l'organizzazione dei file sul server , mentre Socket.io[14] garantisce l'invio dei comandi dal client al server come intuibile dal nome tramite socket.

2.6 Il progetto di partenza

Il modulo di partenza "ardrone-nodejs-browser-control"[15], Implementato da Rohit Ghatol[16] e pubblicato nel suo blog nel 2013 a scopo illustrativo, è un progetto abbastanza semplice che si occupa di inviare alcuni dei comandi base al drone, tramite un client html, ricevendo uno stream del video tramite la libreria dronestream, così si è deciso di modellare il tutto su questa struttura aggiungendo comandi ed eliminando la parti non necessarie. Il server che sostanzialmente rimarrà invariato è scritto in poche righe di codice, ed è strutturato in due moduli che si occupano di gestire i comandi e lo streaming del video.
Il client (Figura 12) di questo progetto, scritto in html, javascript e css mostra quattro bottoni per l'invio di alcuni comandi base, una barra per lo stato della batteria, il video streaming dal drone e il video della webcam del pc. I collegamenti tra client e server sono realizzati tutti tramite la libreria socket.io ricevendo il video alla porta 3001, ed inviando i comandi da far eseguire alla porta 3002 del server che gira in locale.

13 http://expressjs.com/it/
14 http://socket.io/
15 https://github.com/rohitghatol/ardrone-nodejs-browser-control
16 http://www.rohitghatol.com/

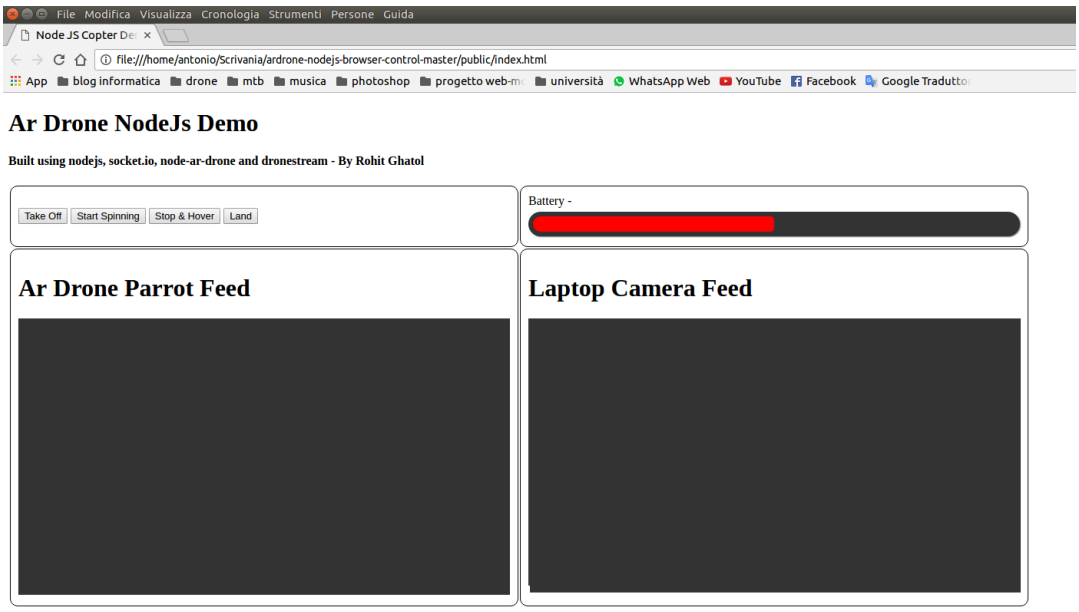

Figura 12 - Client del progetto “ardrone-nodejs-browser-control”

CAPITOLO 3
Architettura di rete

3.1 L’accesso al terminale di Ardrone

Come anticipato nel primo capitolo, Ardrone sfrutta la rete wifi per ricevere i comandi e trasmettere video/dati, come vantaggio di questa scelta c’è la possibilità di poter accedere al computer di bordo tramite un indirizzo ip della rete che si crea, di default questo indirizzo è 192.168.1.1, avendo di conseguenza accesso a tutti i file di configurazione. Da qui il punto di partenza per poter raggiungere lo scopo di collegare il drone alla rete dell’ateneo di Perugia, per estendere la limitata portata del segnale Wi-Fi e controllare il drone da remoto.

3.2 Modifche

Per capire e studiare i modi per effettuare la connessione del drone ad una rete, sono state effettuate una serie di modifiche, quindi è stato introdotto nell’architettura un router, allontanandoci così dal concetto di rete punto a punto sul quale si basa il software di controllo AR.FreeFlight, in modo da avere 3 punti sulla rete e quindi realizzare una connessione pc-router-drone. Disattivate le chiavi di sicurezza del router e cambiato l’indirizzo ip di default dello stesso (da 192.168.1.1 che è l’indirizzo assegnato di default all’ ardrone in 192.168.1.254) il collegamento è avvenuto tramite la configurazione dell’interfaccia wireless del drone. Quindi una volta eseguito il comando telnet all’indirizzo del drone, è stato riavviato il server dhcp ed è stata configurata l’interfaccia ath0

indirizzando il router come gateway di default (Figura 13), così collegando il computer al router si può accedere al drone senza problemi.

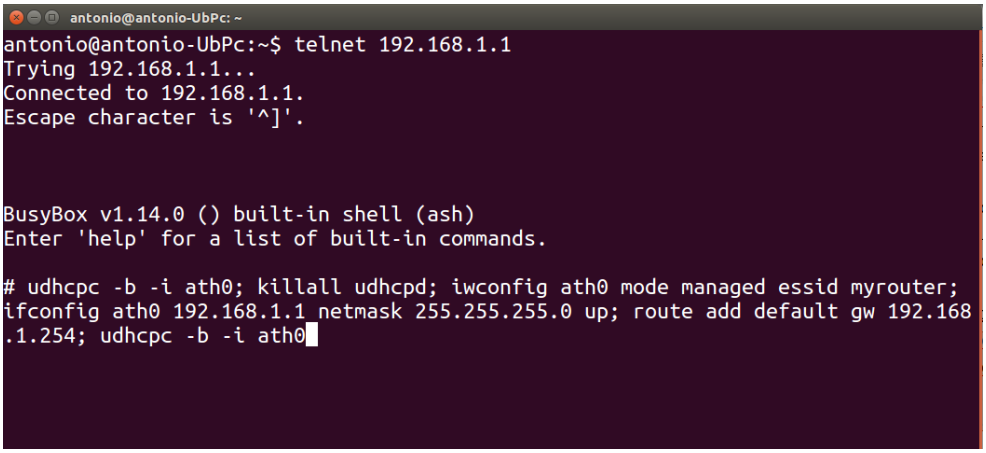

Figura 13 – connessione del drone al router
"myrouter" non protetto

Secondo passo è stato collegare il drone ad un router protetto, inizialmente da chiave wep, successivamente da wpa2 per il quale è stato trovato uno script[17] dal quale si è preso lo spunto per il punto successivo, la connessione alla rete dell'università di Perugia.

Per estendere la portata del wifi in tutto l'ateneo si è cercato il modo per poter connettere il drone alla rete unipg-802.1x con le credenziali d'accesso degli studenti, operazione resa possibile dalla collaborazione degli amministratori di rete che hanno aggiunto alcune eccezioni al firewall, ed effettuando la connessione tramite wpa_supplicant[18], già presente nella versione linux del drone. Questo componente IEEE 802.1X/WPA utilizzato dai client, implementa la negoziazione della chiave con un WPA Authenticator, controlla il roaming e l'associazione/autenticazione del driver wireless.

17 https://github.com/daraosn/ardrone-wpa2

18 https://wiki.archlinux.org/index.php/WPA_supplicant_(Italiano)

Il file di configurazione si trova in /etc/wpa_supplicant.conf e tramite “vi”, unico editor presente, deve essere aggiornato (Figura 14),

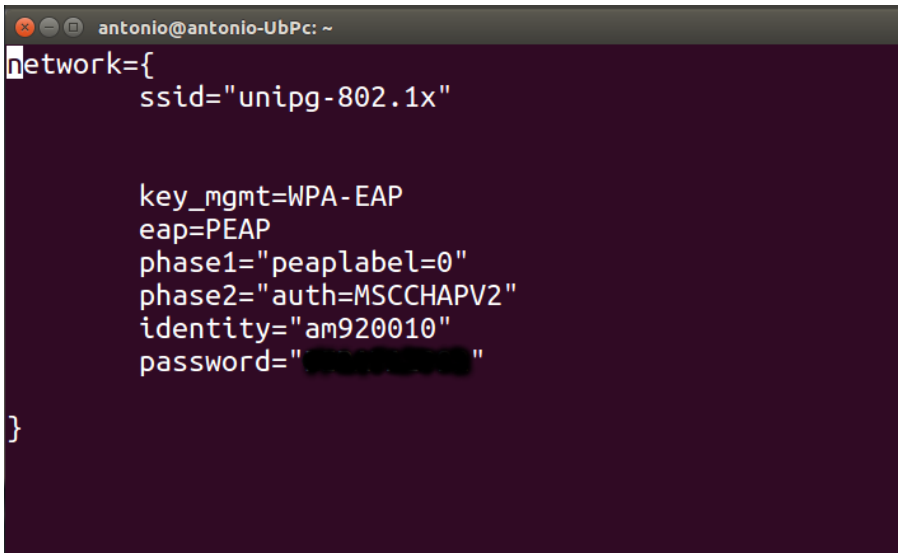

Figura 14 - File di configurazione wpa_supplicant.conf

e come in precedenza è necessario riavviare il dhcp e riconfigurare l’interfaccia in modo che legga i parametri del file di configurazione wpa_supplicant.conf (Figura 15).

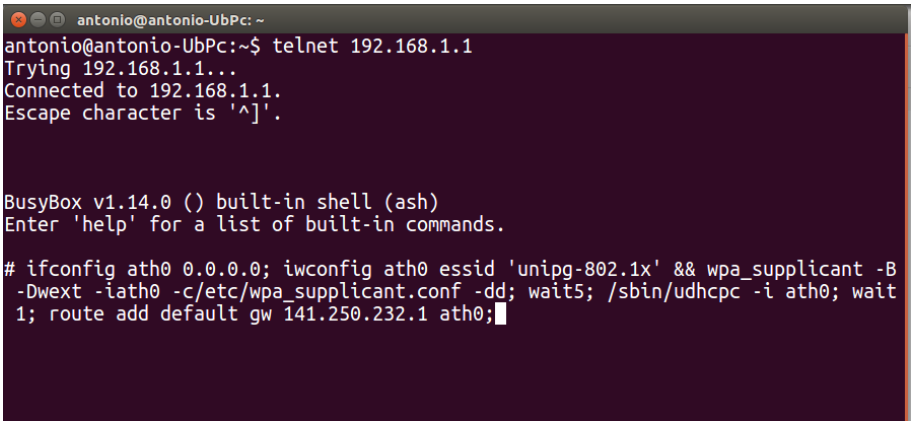

Figura 15 – connessione con wpa_supplicant

Il drone prenderà sempre l’indirizzo ip 141.250.232.5 assegnato “staticamente” dal dhcp dell’università, unica accortezza per evitare di finire nelle blacklist del wifi unipg è aggiornare la data del sistema operativo del drone che tende a modificarsi dopo ogni spegnimento.

3.3 Client-server-ardrone

A questo punto, collegato il drone alla rete si è pensato alle soluzioni per pilotarlo da remoto e ricevere le immagini trasmesse dalla camera sul browser, così dopo varie ricerche si è deciso di introdurre un server per la gestione dei comandi e un' interfaccia web in html per il client (Figura 16).

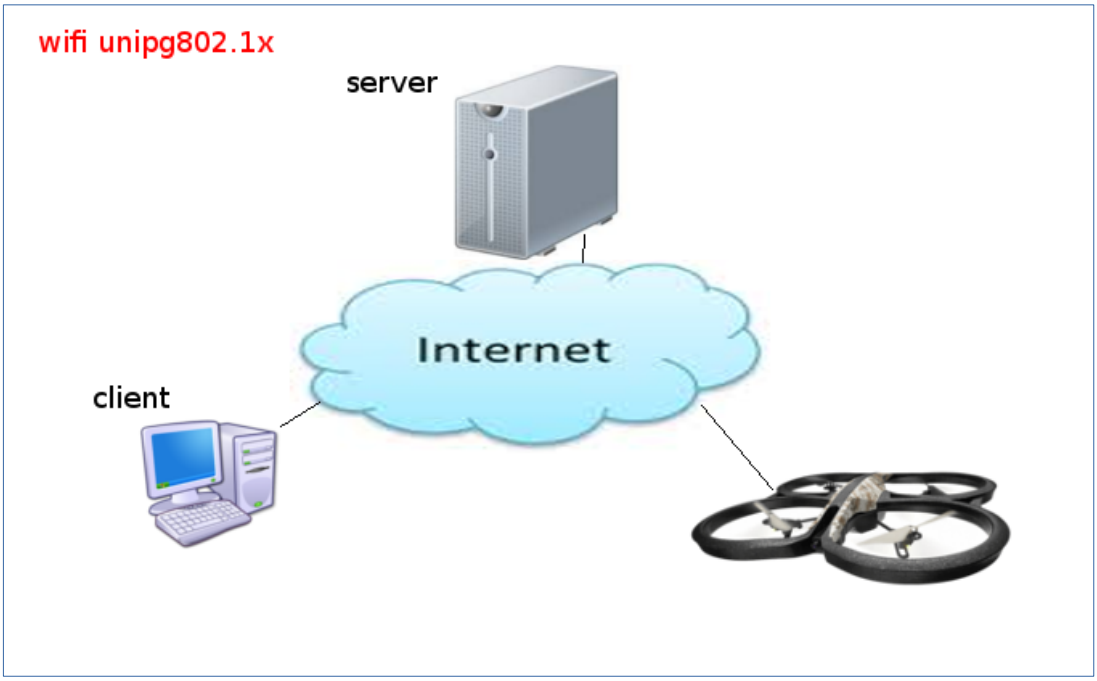

Figura 16 – client-server-ardrone

CAPITOLO 4
Autonomus Ardrone

4.1 Il server

Dopo lo studio di tutti i moduli utili per la realizzazione del progetto si è intervenuti modificando alcune parti per adattarle allo scopo del progetto, il server sostanzialmente rimane invariato, verrà successivamente introdotto l'indirizzo ip che servirà per raggiungere il server nella rete. Si tiene la struttura modulare e ciò permette di operare solo sul modulo controller.js implementando tutti i comandi che mancano, ottenendo un controllo completo del drone dal browser. Quindi il server richiama il modulo "Express" per la gestione dei file e i due moduli camera-feed.js e controller.js, uno si occupa di eseguire dronestream e quindi lo streaming della videocamera del drone, l'altro richiama la libreria node-ardrone e attraverso socket.io si occupa dei collegamenti con il client, inviando costantemente lo stato della batteria e ricevendo all'occorrenza i comadi da inviare al drone.

4.2 Il client

Dopo una divisione del codice html, javascript e css, dal client(Figura 17) è stata eliminata la parte che riguardava lo streaming della webcam del pc, sostituita dal canvas che permette al drone di effettuare il tracking di un colore di cui si parlerà successivamente, quindi si è intervenuti nell'aggiunta dei button con le annesse funzioni javascript che inviano i comandi al server, più il button che lancia la funzione tracking del colore e consente al drone di seguirlo in volo.

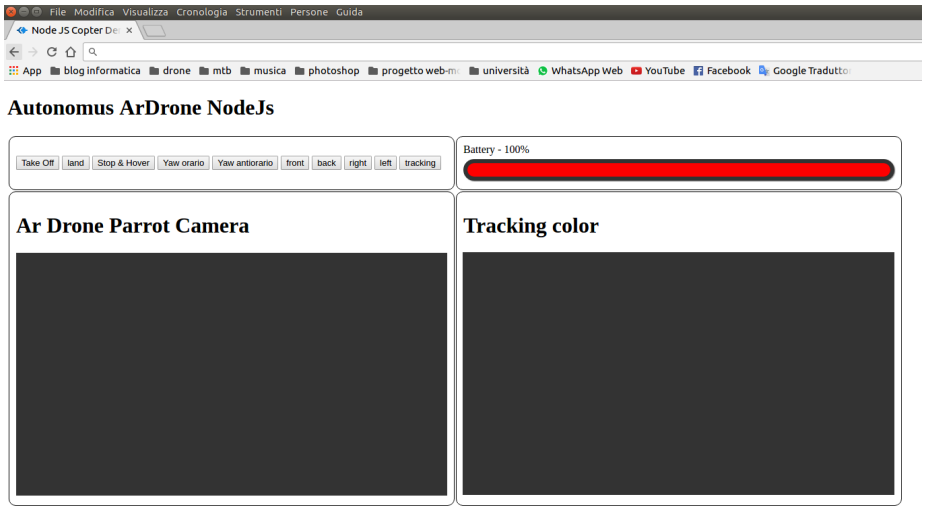

Figura 17 - Client modificato

4.3 Lo scopo originario

Il progetto originario di questa tesi includeva la realizzazione di una funzione che riuscisse a fare in modo che il drone riconoscesse un determinato oggetto e lo seguisse in volo, ma effettuando lo studio per il riconoscimento dell'oggetto si è raggiunta la consapevolezza di dover lavorare con gli haar cascade file[19] della libreria opencv[20]. La creazione di un file di questo tipo avrebbe impiegato settimane di computazione, poiché per crearli si usa un complesso meccanismo di analisi di moltitudini di immagini positive (che contengono l'oggetto da riconoscere) e negative (che non contengono l'oggetto). Sono state effettuate prove con circa 1200 immagini, ma dopo giorni di computazione del cascade classifier Training[21], non sono stati raggiunti risultati soddisfacenti, inoltre i molti esempi trovati online si occupano esclusivamente del riconoscimento del viso, possibilità scartata per la potenziale pericolosità delle eliche dell'ardrone. Durante lo studio sono state trovate molte librerie che si occupano del riconoscimento delle immagini, ma fra tutte si

19 http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html
20 http://opencv.org/
21 http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html

è scelto di analizzare quelle realizzate per javascript, così la rosa è stata ristretta a jsfeat[22] e trackingjs. Entrambe librerie valide e formite di un' ottima documentazione, la prima eccelle nel riconoscimento di punti, angoli e elaborazione dell' immagine, la seconda sulla quale è ricaduta la scelta presenta molte funzioni per il riconoscimento delle immagini, di conseguenza si è arrivati alla conclusione di utilizzare le funzioni per il tracciamento del colore; interessante fra tutti si era mostrato il file hand.js che si occupa del riconoscimento delle mani, rivelandosi però inutile per le molte imprecisioni.

4.4 Tracking.js

Ultimo passo dunque è lo studio della libreria Tracking.js[23], fornita di una buona documentazione[24] e molti esempi, presenta degli algoritmi che eseguono il tracciamento del colore in real-time, richiamati nel codice javascript attraverso il costruttore tracking.ColorTracker(), che ha come parametri i colori che si desidera tracciare; il costruttore restituisce un oggetto(che chiameremo tracker) nel quale è possibile stabilire tutto ciò che si deve eseguire una volta riconosciuto il colore ed infine si lega l'oggetto restituito all'elemento html (<img>, <video> e <canvas>) in cui si desidera eseguire il tracking, unico problema: la libreria non supporta webgl.

22 https://inspirit.github.io/jsfeat/
23 https://trackingjs.com/
24 https://trackingjs.com/docs.html

4.5 Il problema

Dal momento che la libreria tracking.js non supporta webgl, ci troviamo di fronte ad un grosso problema e analizzandolo, ci si accorge che la libreria dronestream che si occupa della trasmissione del video nel browser, riporta il video su una webgl canvas, inutili sono i tentativi di trasferire il tutto su un tag <video> poiché come detto in precedenza questo tipo di tag non legge il formato video h.264 baseline. Così si è pensato alla soluzione di tornare alle funzioni della libreria node-ardrone, così il flusso video tramite socket.io viene letto come una serie di immagini png e perciò inserito su un tag <img> aggiornato constatemente al quale è possibile legare il tracker. Il tutto funziona, ma come detto nel capitolo 2 le funzioni che si occupano dello stream sono affette da un problema di delay di 4 secondi che le rende inutilizzabili, rendendo anche impossibile il controllo da remoto del drone.

4.6 La soluzione

Accantonata così l'idea di usare le funzioni "standard", si è pensato ad un modo per aggirare il problema, continuando ad usare dronestream, cercando un modo di recuperare dalla webgl canvas le immagini trasferendole su un canvas dove è possibile eseguire il tracking. Così nel codice javascript è stata aggiunto un id al canvas creato dalla libreria dronestream e copiare il flusso di immagini in un nuovo canvas su cui è applicabile il tracking, il tutto è eseguito dalla funzione denominata copyCanvas()(Figura 18).

```
function copyCanvas(){
    var gl =document.getElementById("stream");
    var glcont=gl.getContext('webgl');
    var canvas2d = document.getElementById('workCanv');
    canvas2d.src = gl.toDataURL();
}
```

Figura 18 - copyCanvas()

La funzione viene richiamata ad intervalli regolari di pochi millisecondi in modo da rendere il delay sull'immagine quasi impercettibile dall'occhio umano, non causando alcun problema per il controllo da remoto.

Conclusioni e applicazioni future

Riassumendo, dunque i punti focali di questa tesi sono stati il collegamento del drone ad una rete wifi per aumentarne la portata, e il tracking autonomo di un oggetto una volta riconosciuto un colore, quindi pensando alle applicazioni future, si dovrà tenere conto del riconoscimento delle forme di un determinato oggetto da seguire, quindi anche una persona, mentre per ciò che riguarda la rete, si potrebbe pensatre al collegamento di un modulo 3g che opportunamente configurato alla porta usb dell' ardrone, potrebbe rendere la portata del segnale dell'ardrone "illimitata" e ciò consentirebbe molte applicazioni, tenendo conto come unico limite della durata delle batterie.
Purtroppo la serie ardrone della parrot è stata "abbandonata", lasciando il posto alla serie bepop che cambiando design introduce nuovo hardware e una nuova fotocamera in grado di effettuare riprese a 1080p, tuttavia grazie alla compatibilità dei moduli il progetto dovrebbe funzionare, con le dovute modifiche, anche sulla nuova serie.

Appendice

Da non dimenticare

- L’hardware necessario per eseguire questo progetto deve essere dotato di una memoria ram di almeno 6GB, sia per quanto riguarda il server e sia per il client, a causa della grande mole di dati che viene inviata durante lo streaming delle immagini.

- Installare nodejs e i relativi moduli del progetto.

-Ricordare di inserire l’indirizzo IP del server nei moduli nodejs: dronestream, socket.io.

- Ricordare di inserire l’indirizzo IP dell’ardrone nel modulo nodejs: ardrone.

- Controllare che le connessioni tra server, ardrone e client non siano intercettate dai firewall, in tal caso bisogna aggiungere le regole tcp,udp per consentire il funzionamento dell’architettura.

IL CODICE

server.js

```
var express = require('express')
    , app = express()
    , server = require("http").createServer(app)
app.use(express.static(__dirname + '/public'));
require("./drone/camera-feed");
require("./drone/controller");
app.listen(3000,"141.250.5.252");
```

camera-feed.js

```
require("dronestream").listen(3001);
```

controller.js

```
var io = require('socket.io').listen(3002);
var arDrone = require('ar-drone');
var client = arDrone.createClient();
io.sockets.on('connection', function (socket) {
   setInterval(function(){
        var batteryLevel = client.battery();
        socket.emit('event', { name: 'battery',value: batteryLevel});
    },1000);

    socket.on('event', function (data) {
        if(data.name=="takeoff"){
            console.log("Browser asked Ar Drone to Take Off");
            client.takeoff();
        }
        if(data.name=="stop"){
            console.log("Browser asked Ar Drone to Stay and Hover");
            client.stop();
        }
```

```
        if(data.name=="land"){
            console.log("Browser asked Ar Drone to Land");
            client.land();
        }
          //aggiungo comandi
          if(data.name=="left"){
            console.log("Browser asked Ar Drone to left");
            client.left(0.1);
        }
          if(data.name=="right"){
            console.log("Browser asked Ar Drone to right");
            client.right(0.1);
        }
          if(data.name=="clockwise"){
            console.log("Browser asked Ar Drone to cwise");
            client.clockwise(0.1);
        }
          if(data.name=="counterClockwise"){
            console.log("Browser asked Ar Drone to ccwise");
            client.counterClockwise(0.1);
        }
          if(data.name=="front"){
            console.log("Browser asked Ar Drone to front");
            client.front(0.1);
        }
          if(data.name=="back"){
            console.log("Browser asked Ar Drone to back");
            client.back(0.1);
        }
    });
});
```

index.html

```
<!DOCTYPE html>

<html>
	<head>
	<title>Node JS Copter Demo</title>
	<script src="nodecopter-client.js"></script>
	<script src="socket.io.js"></script>
	<script src="jquery.min.js"></script>
	<script src="tracking-min.js"></script>
	<link rel="stylesheet" type="text/css" href="style.css">
	<script type="text/javascript" src="funzioni.js"></script>
  </head>
	<body>
		<h1>Autonomus ArDrone NodeJs</h1>
		<table>
		<tr>
		<td >
			<button id="takeoff">Take Off</button>
			<button id="land">land</button>
			<button id="stop">Stop & Hover</button>
			<button id="clockwise">Yaw orario</button>
			<button id="counterClockwise">Yaw antiorario</button>
			<button id="front">front</button>
				<button id="back">back</button>
			<button id="right">right</button>
			<button id="left">left</button>
			<button id="tracking">tracking</button>
		</td>
		<td>
					<span>Battery</span> - <span id="battery-
value"></span>
				<div class="bar color0">
				<span id="battery-indicator"
style="width:50%"></span>
				</div>
		</td>
		</tr>
		<tr>
		<td>
			<h1>Ar Drone Parrot Camera</h1>
			<div id="placeholder"> </div>
		</td>
		<td>
			<h1>Tracking color</h1>
```

```
                    <canvas id="canvas" width="640" height="360"></canvas>
                        </td>
                </tr>
                </table>
                <img id="workCanv" width="640" height="360">
        </body>
</html>
```

funzioni.js

```
 $(function () {
        var socket = io.connect('http://141.250.5.252:3002');
         socket.on('connect', function () {
                 console.log("Connection Successful");
                                        });
         socket.on('event', function (data) {
                                                if(data.name=="battery"){
                    $("#battery-indicator").css('width',data.value+'%');
                    $("#battery-value").html(data.value+'%');
                }
         });
        var start =0;
        var startad=0;
        var cTakeoff= false;
        var control = false;
        var controlad = false;
         function track(){
                var img = document.getElementById('workCanv');
                                var canvas = document.getElementById('canvas');
                                var canvasWidth  = 640;
                var canvasHeight = 480;
                ctx = canvas.getContext('2d');
                ctx.fillStyle = "rgb(0,255,0)";
                ctx.strokeStyle = "rgb(0,255,0)";
                ctx.drawImage(img, 0, 0, canvasWidth, canvasHeight);
                var tracker = new tracking.ColorTracker(['cyan']);
                tracker.on('track',function(event){

        event.data.forEach(function(rect){

        draw(rect.x,rect.y,rect.width,rect.height,ctx);
                                                                console.log("x:"+
rect.x);
                                                                console.log("y:"+
rect.y);
                                                                console.log("w:"+
rect.width);
```

```
                                                    console.log("h:"+
rect.height);
                                                    //inizializzazione
centro per iniziare a seguire il colore
                                                    //avanti dietro
                                                    if(rect.width>90 &&
rect.width<120 && cTakeoff && controlad==false){
        socket.emit('event',{name:"stop"});
                                                            startad++;
                                                            controlad =
true;
        console.log(startad);
        console.log("centroad");
        console.log("takeoff:"+cTakeoff);
                                                    }
                                                    if(startad!=0){
        console.log(controlad+"startad");
        if(rect.width<90 && controlad){
        console.log(controlad+"front");
        socket.emit('event',{name:"front"});
        controlad = false;
        startad=0;
                                                            }
                                                            else{
        if(rect.width>120 && controlad){
        console.log(controlad+"back");
        socket.emit('event',{name:"back"});
        controlad=false;
        startad=0;
```

```
                                                                    }
                                                                }
                                                            }
                                                            //destra sinistra

        console.log(control);
                                                            if(rect.x<400 &&
rect.x>200 && cTakeoff && control==false){

        socket.emit('event',{name:"stop"});
                                                                start++;
                                                                control =
true;

        console.log(start);

        console.log("centro");

        console.log("takeoff:"+cTakeoff);
                                                            }
                                                            console.log(start);
                                                            if(start!=0){


        console.log(control+"start");

        if(rect.x<200 && control){

        console.log(control+"left");

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

        control = false;

        start=0;

                                                                }
                                                                else{


        if(rect.x>400 && control){

        console.log(control+"right");

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

        control=false;
```

```
        start=0;
                                                                    }
                                                                }
                                                            }
                                                        });
                    });
        function draw(x, y, w, h, ctx) {
                            console.log('draw');
                    ctx.strokeRect((x)|0,(y)|0,(w)|0,(h)|0);
        }
        tracking.track('#canvas',tracker);
    }//fine funzione track

    function startArDRoneStream() {
        new NodecopterStream(document.getElementById("placeholder"), {port:
3001});
         var gl = document.getElementById("placeholder").children[0];
         gl.id="stream";
    }
    function copyCanvas(){
        var gl =document.getElementById("stream");
         var glcont=gl.getContext('webgl');
         var canvas2d = document.getElementById('workCanv');
         canvas2d.src = gl.toDataURL();
    }
    function startArDroneController(){
        $("#takeoff").click(function(){
                    console.log("Asking Server to send takeoff command to Ar
Drone");
                socket.emit('event',{name:"takeoff"});
                cTakeoff = true;
        });
         $("#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"});
              cTakeoff = false;
         });
         $("#clockwise").click(function(){
              console.log("Asking Server to send clockwise command to Ar
Drone");
              socket.emit('event',{name:"clockwise"});
```

```
        });
        $("#counterClockwise").click(function(){
            console.log("Asking Server to send counterClockwise command to Ar
Drone");
            socket.emit('event',{name:"counterClockwise"});
        });
        $("#front").click(function(){
            console.log("Asking Server to send front command to Ar Drone");
            socket.emit('event',{name:"front"});
        });
        $("#back").click(function(){
            console.log("Asking Server to send back command to Ar Drone");
            socket.emit('event',{name:"back"});
        });
        $("#right").click(function(){
            console.log("Asking Server to send right command to Ar Drone");
            socket.emit('event',{name:"right"});
         });
         $("#left").click(function(){
            console.log("Asking Server to send left command to Ar Drone");
            socket.emit('event',{name:"left"});
         });
         $("#tracking").click(function(){
            console.log("start tracking");
            setInterval(function(){copyCanvas();
                                                    track();},200);
                                        });
        }
    startArDRoneStream();
    startArDroneController();
});
```