**Intro**

So far we’ve used NodeJS and some basic JavaScript to control the drone, however this is a bit static – what if we want to perform different commands during flight? Or what if we want to view a live feed from the drone? This is where Webflight comes in.

Webflight is a great open-source project provided by Laurent Eschenauer (<https://github.com/eschnou>), the repo can be found here (<https://github.com/eschnou/ardrone-webflight>). This program allows the user to manually control the drone using their computer keyboard while viewing a live feed through the web browser and comes with several interesting plugins.

**Install Webflight**

Scrolling down a bit on the Webflight github page you’ll see the below instructions for installation. The yellow highlighted line has been added by me as I found that bower wasn’t recognised by the computer without this (bower install will fail, unknown command). Run these commands in a command prompt window in a location of your choice, I personally used the git bash window I’m using for version control to complete the below and it worked fine.

git clone https://github.com/eschnou/ardrone-webflight.git

cd ardrone-webflight

npm install

npm install bower –g additional step as it doesn’t recognise bower without

bower install

The three screenshots below demonstrate the process of installing everything, and installing bower using npm, using the –g tag to set a global path variable to it as well (this just means that you can run the command ‘bower’ anywhere and it’ll be recognised, rather than having to run it from the directory it lives in).

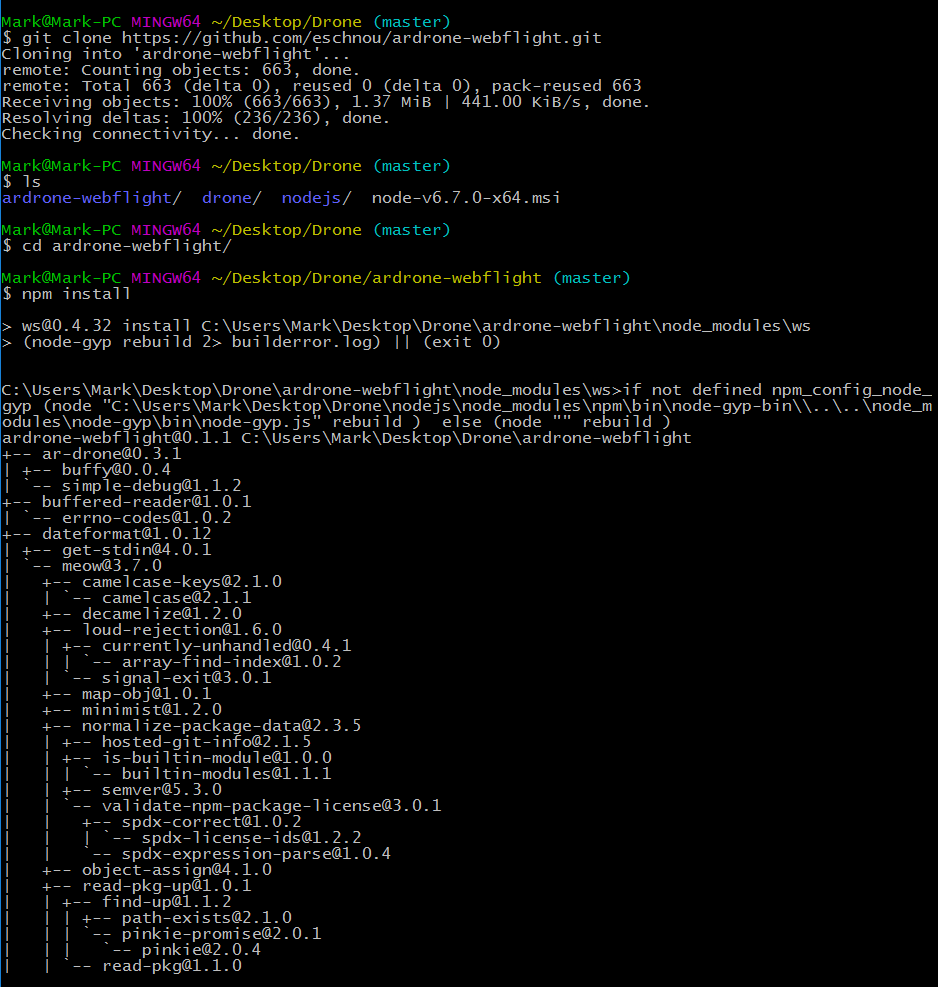


Figure 1: Cloning repo from GitHub and using npm install

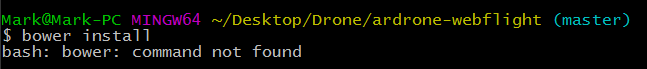


Figure 2: At first your computer will not understand what bower 'is'

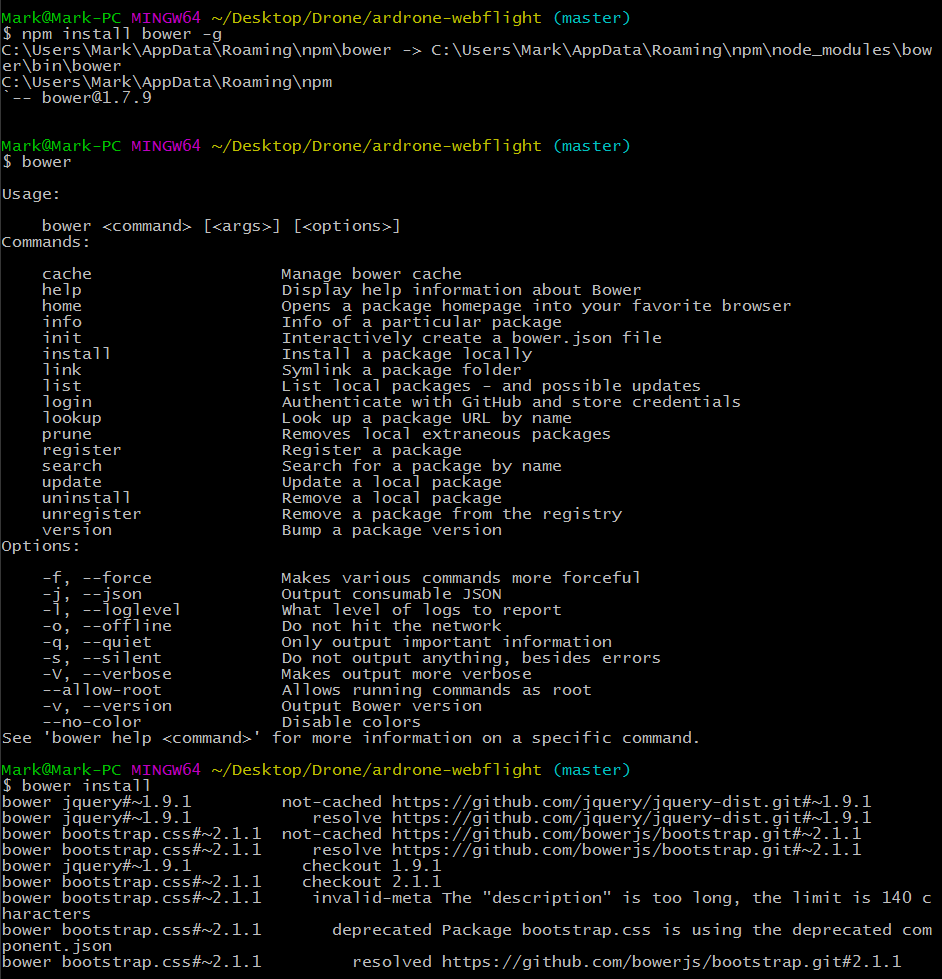


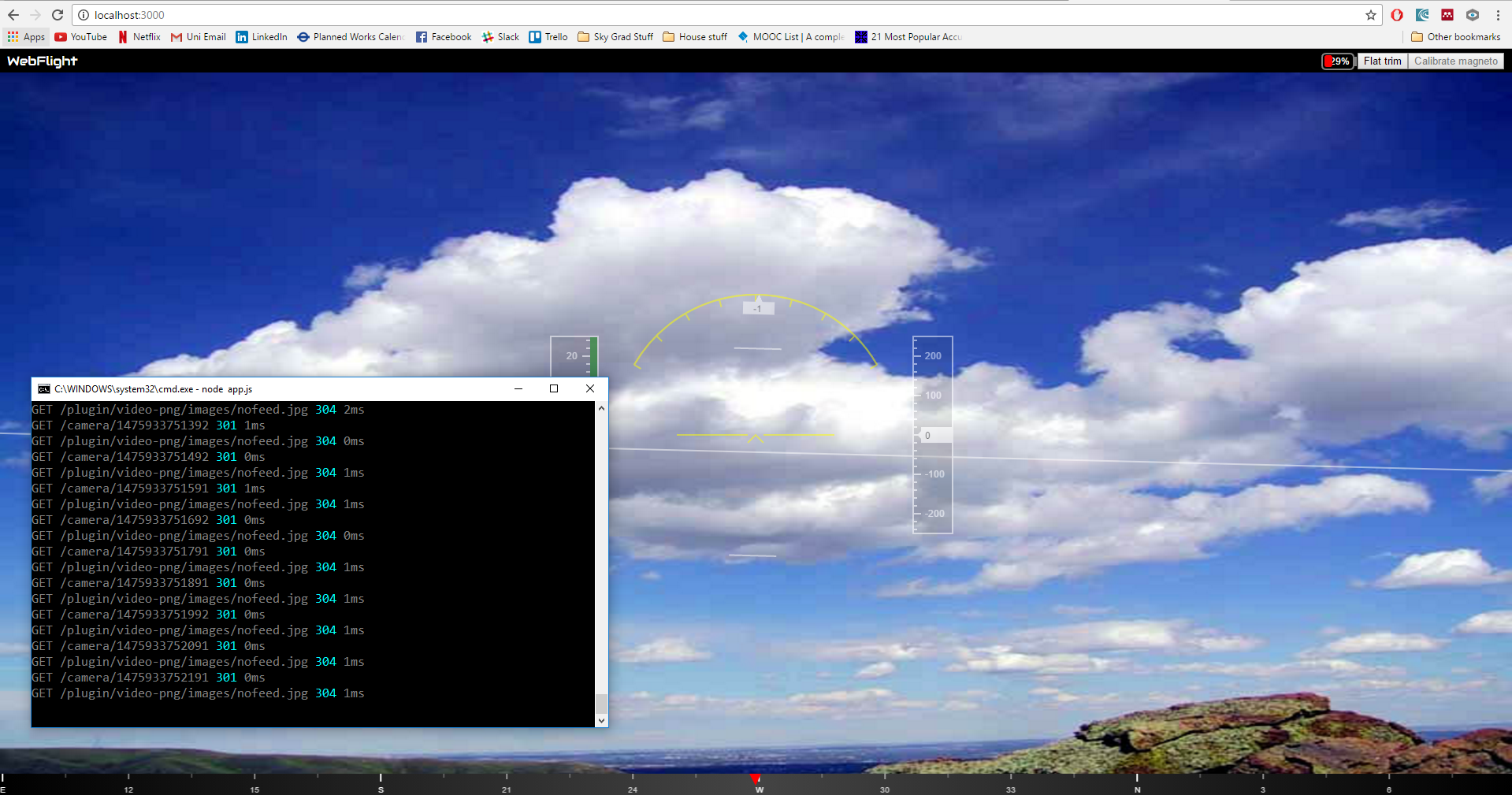
Figure 3: Use npm to install bower and use the -g tag to set it to global path, bower is now recognised and you can run bower install

**Configuration**

Now that everything is installed we need to make sure that Webflight is configured as we like it. To get a basic level of operation running we won’t need to do much so we’ll do the bare minimum for now to get up and running.

If you navigate to the ardrone-webflight folder that you recently cloned from GitHub, you’ll find a file called config.js.EXAMPLE – simply put this is your configuration file that tells Webflight which plugins to use etc. For now, rename the file to config.js (remove the ‘.EXAMPLE’ bit at the end), so as it’s recognised as the correct file for Webflight to use. Aside from config.js, there is one other file in the same directory that is very important, that being app.js. App.js is the main, central JavaScript program that includes links to the config.js file (hence why renaming this a minute ago was important!) as well as basic controls etc. There’s no need to edit this right now, that’s the basic configuration done!

If you now open your command prompt in the ardrone-webflight directory and type ‘*node app.js’* in you will run the Webflight application! At the moment you will see a couple of errors to do with ffmpeg not being installed, for now don’t worry about that, simply open up your web browser and type in localhost:3000 in the URL bar at the top – you should see the following.



You’ll notice that your drone is (likely) not pointed at a nice view of clouds and whatnot, this is a stock image if you haven’t already guessed. The reason it displays this stock image is due to FFMPEG not being installed, this program is used to render images and videos from the drone so until we get that set up, we’re stuck staring at this!

However, it’s not all bad! We’ve got a nice HUD (Heads Up Display) including a compass along the bottom of the screen and some nice subtle instruments in the centre of the screen.

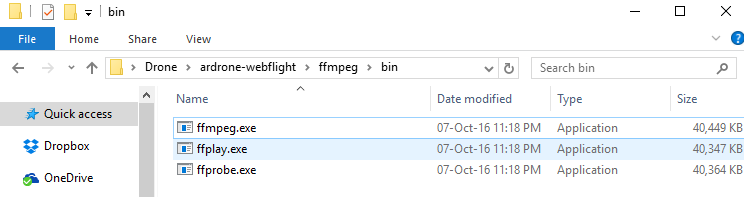
FFMPEG (FFS)

Downloaded FFMPEG (latest static 64-bit release) from here <https://ffmpeg.zeranoe.com/builds/>

Put it inside the ardrone-webflight folder and extracted it (right click, extract here). Renamed the extracted folder to ‘ffmpeg’

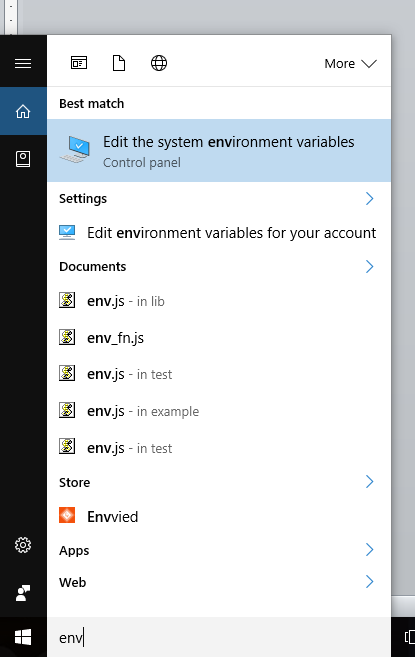
Good guide here <http://www.wikihow.com/Install-FFmpeg-on-Windows>

Within the ffmpeg folder you’ll find a folder called ‘bin’, enter this folder and copy the location of this. To do this, click on the address bar (the bit that says ‘*Drone -> ardrone-webflight > ffmpeg > bin’* in the image below) and copy the path that shows up.

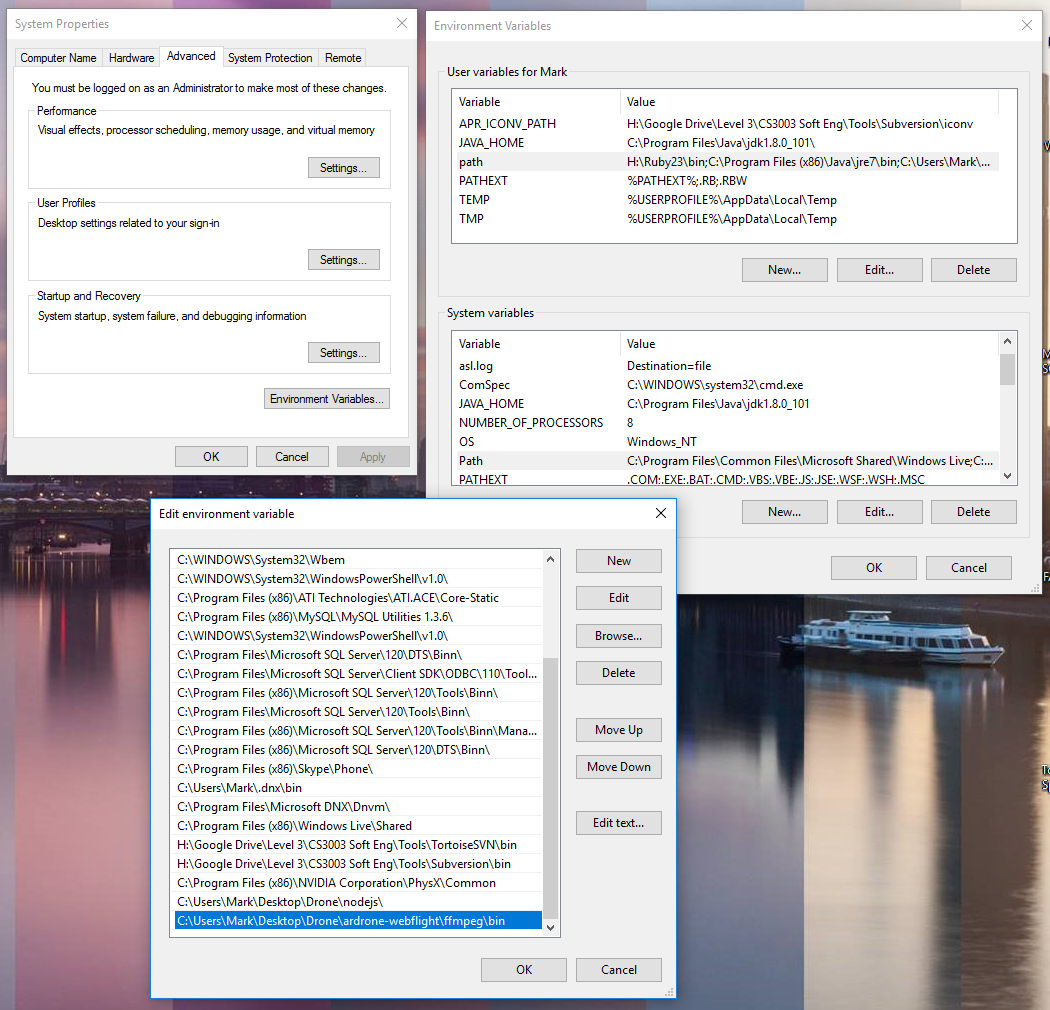


Now we need to edit the environment variables for the computer so as you can use ffmpeg from anywhere. It would make sense that you could use the –g tag like we did earlier for bower, but I haven’t figured out why that doesn’t work in this case so I’m just putting it down to FFMPEG being awkward and doing it the good old fashioned way.

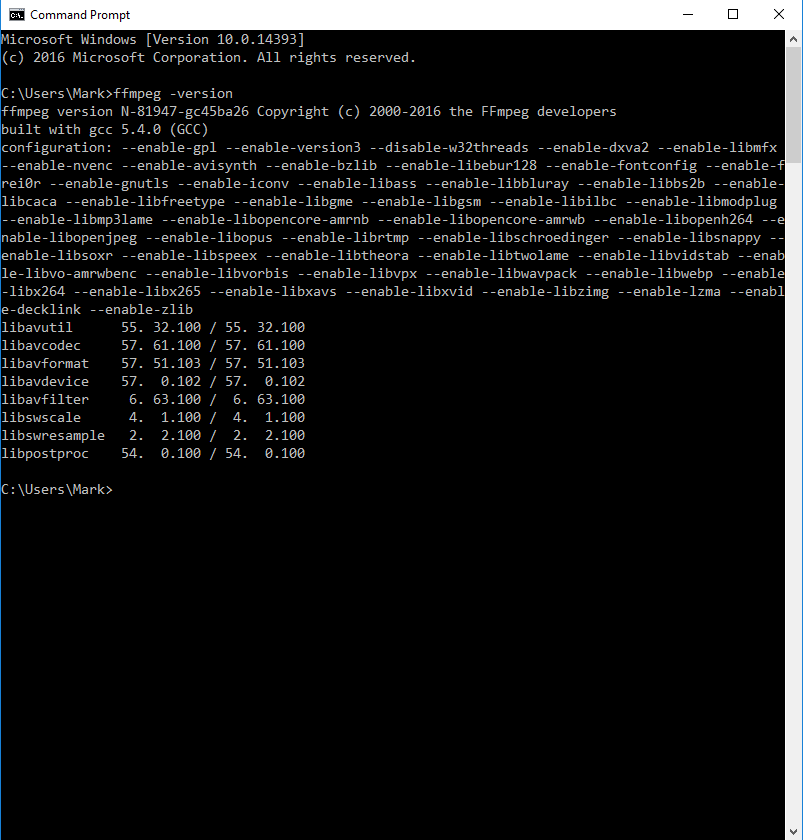
I’m using windows 10, so if you are too, all you need to do is start typing *‘env’* in the ‘Ask me anything’ bar in the bottom left of your screen (Cortana?), you should see an option to edit the system environment variables as shown below, click this.



You should now be presented with the top left window shown in the screenshot below, click the Environment Variables… button to be taken to open the top right window from the screenshot below. Click on the Path item (PATH, path – whichever way it’s shown on your computer), then click on the Edit button to reach the window shown at the bottom of the screenshot below. Click New and then paste in the path you copied earlier from the ffmpeg/bin folder.



Now if we close any command prompt windows and reopen them, typing *‘ffmpeg –version’* you should see an output similar to below, if so, ffmpeg is installed and ready to be used with Webflight!



If you now navigate back to your ardrone-webflight folder and run node app.js in command line, you should notice that the ffmpeg errors have gone and if you open localhost:3000 in your browser you are receiving a feed live from your drone! Below is my feed while writing this guide, the drone is currently looking out of the window enjoying a view of perfect English summer weather.

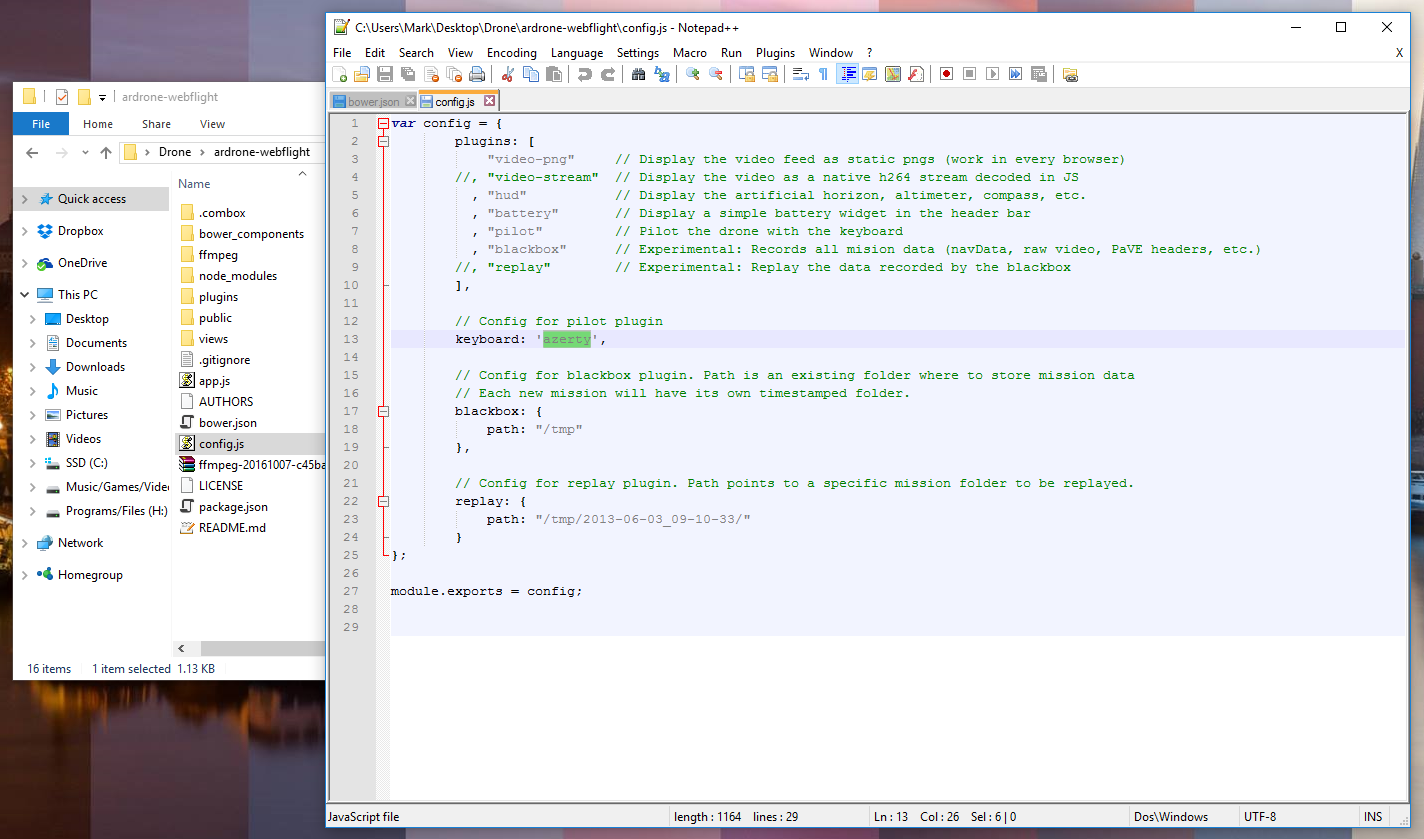


**Controlling the drone using Webflight**

Seeing through your drone is all good and well but what about flying it? As long as your drone has at least 20% battery life (see top left of Webflight display in your browser), you can press ‘T’ on your keyboard to take off and ‘L’ to land!

However you might notice that other commands seem to be missing. The up and down arrows seem to make drone go up and down, but what about forwards and backwards? By default the config.js file is set to use a different keyboard layout from QWERTY (UK/US layout) so we just need to change one line in there first.

Open the config.js file, you’ll notice on line 13 there is a keyboard variable set to *‘azerty’*, simply change this to *‘qwerty’* to use Webflight with a standard UK/US keyboard layout. Save and exit the file and restart Webfight for the changes to take effect (close browser window/tab running Webflight and use ctrl+c to stop it in the command prompt window – then re-run node app.js in command prompt and reopen localhost:3000 in your browser).



Once you’ve reconfigured Webflight in the config.js file and started it up again, you can run the following commands in the browser window:

* W – Forward (pitch)
* A – Left (roll)
* S – Backward (pitch)
* D – Right (roll)
* Up – Increase height
* Down – Decrease height
* Left – Rotate left (yaw)
* Right – Rotate right (yaw)
* F – Flip (make sure you’ve got lots of room around the drone!)