# Programming the Parrot AR 2.0 Drone – Beginners Guide 2

In the previous guide (<https://mveeprojects.wordpress.com/2016/10/15/flying-the-ar2-0-drone-beginners-guide-1/>) we used NodeJS and some basic JavaScript to control the drone, however this is all a bit static – what if we want to control the drone in real-time during flight, rather than having to predefine a route for it to fly? Or what if we want to view a live feed from the drone? This is where Webflight comes in. Webflight is a superb open-source project created by Laurent Eschenauer (<https://github.com/eschnou>). Webflight can be downloaded/cloned for free from Laurent’s GitHub repository here (<https://github.com/eschnou/ardrone-webflight>).

# What are we going to do?

1. Go over thenewboston tutorial on installing Git bash
2. Download (Clone) Webflight and install required components
3. Configure Webflight ready for flight
4. Download and configure a program to render the feed from the drone’s camera as a live stream in our browser (FFMPEG)
5. Start flying our drone in real time using our laptop/PC keyboard!

# Stuff you’ll need

## Same as last time

In this guide you’ll need exactly the same bits and pieces as in the previous guide (<https://mveeprojects.wordpress.com/2016/10/15/flying-the-ar2-0-drone-beginners-guide-1/>) (Win10 laptop/PC with Wi-Fi, drone, Notepad++). You’ll also need to have completed the previous guide to ensure you have NodeJS installed as well.

Aside from that we also need an application called Git installed. We will need to use ‘Git Bash’ in order to complete this guide.

## Git

Git is a piece of software that falls under the bracket of Version Control Software (VCS). Git’s job is to make sure files are up to date as well as keeping a history of the different versions of files up to the current point, just in case you need to revert to an older version.

Going into detail about Git and its uses is a little out of the scope of this guide, but you absolutely need Git installed on your laptop/PC to get Webflight installed correctly. Bucky Roberts over at thenewboston.com has put together a great series of easy to understand video tutorials on getting up and running with Git so I would recommend you go through at least the first tutorial on installing Git (<https://thenewboston.com/videos.php?cat=70>) before continuing. The tutorials after the first one are great if you have a GitHub account and want to understand a bit more about VCS tools and their uses. For the sake of this guide, we only need Git to do some very basic stuff so it’s not totally necessary to know loads about it.

Also, have a look around the tutorials on thenewboston.com, there are absolutely loads of really well put together and clearly explained tutorials on programming. It’s an absolute goldmine of programming knowledge!

# Let’s get to it!

1. Go over thenewboston tutorial on installing Git bash

Make sure you’re familiar with Git (Git Bash) before going any further; this will ensure that you can get everything set up ready to use Webflight! Head over to thenewboston’s Git tutorials here (<https://thenewboston.com/videos.php?cat=70>).

## 2. Download (Clone) Webflight and install required components

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 added to help get Bower installed!)

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).

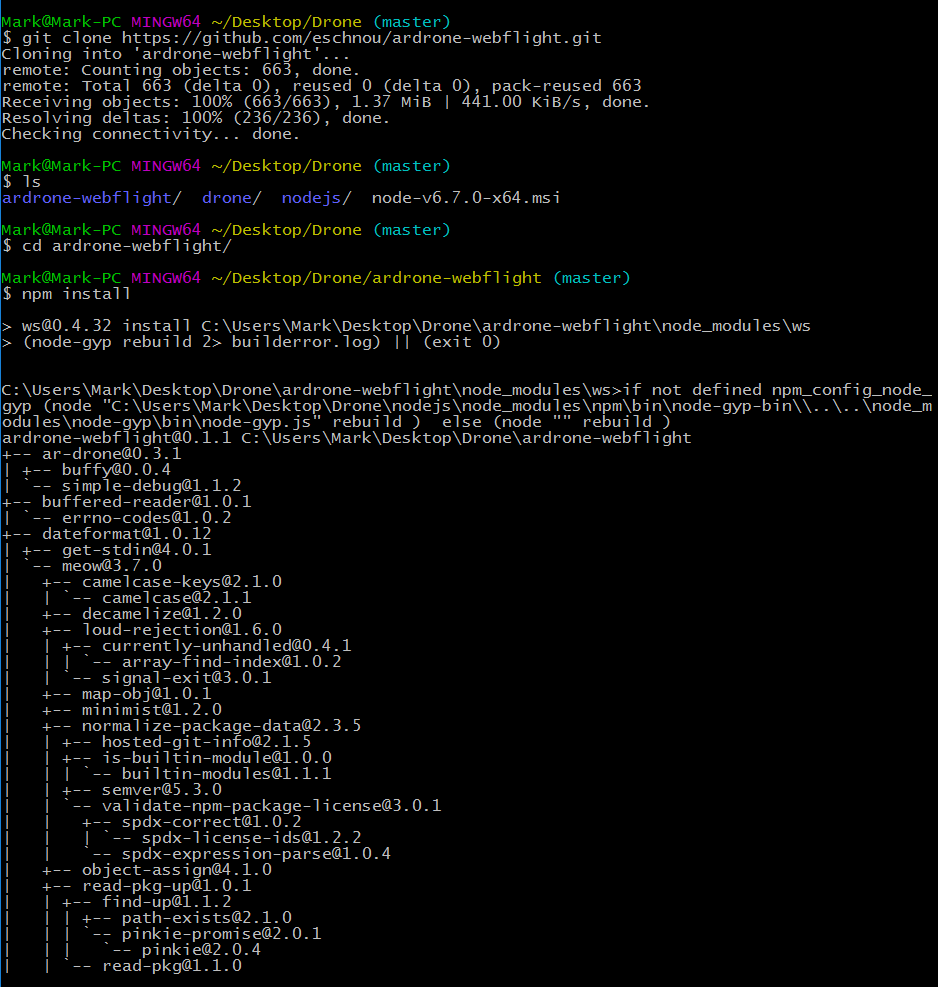


Figure 1: Cloning repo from GitHub and using npm install

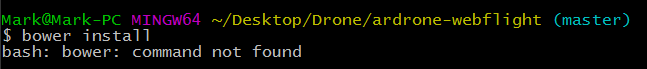


Figure 2: At first your computer will not understand what bower 'is'

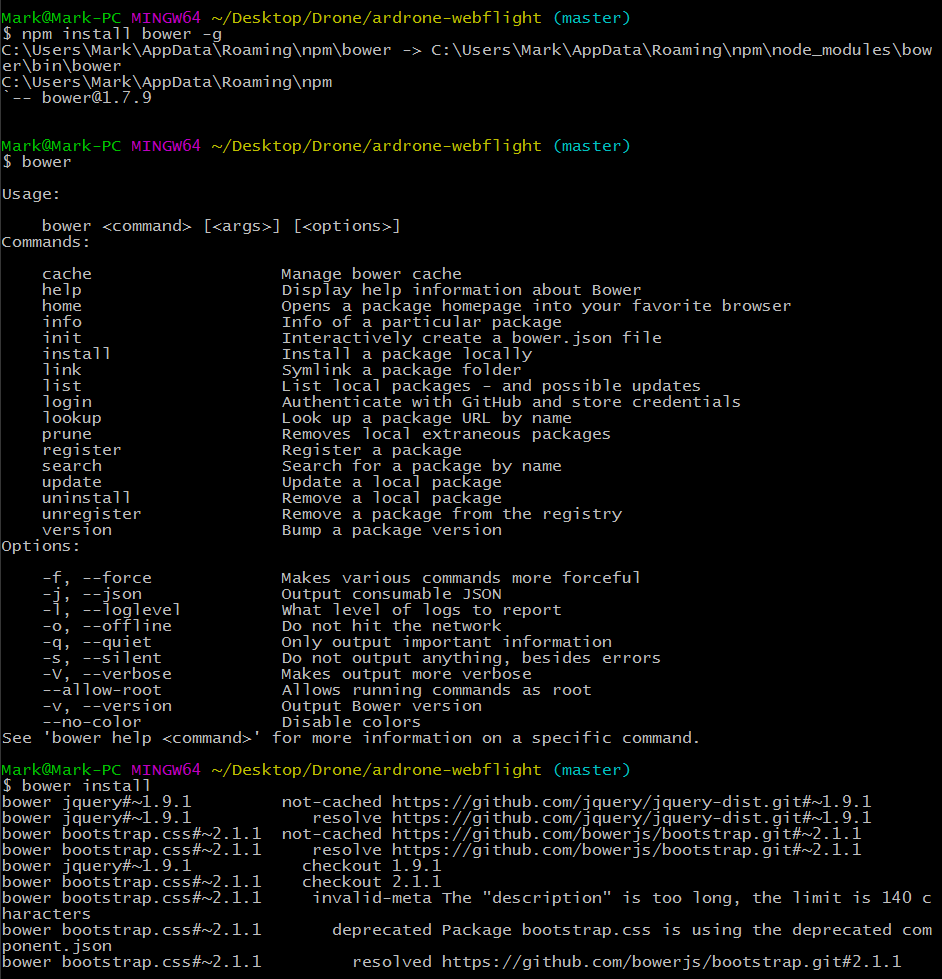


Figure 3: Use npm to install bower and use the -g tag to set it to global path, bower is now recognised and you can run bower install

## Random error I ran into

When testing the installation of Webflight a second time for this guide, I ran into an error when running the *bower install* command (last line). Unfortunately I closed the window before I thought about screenshotting it so I apologise for that. However, part of it included *“exit code #128 fatal”* so if you see that at all, it’s the same error.

While I was googling the error and trying to figure out how to fix it, the gremlin in the system causing this issue magically disappeared and I was able to run the *bower install* command without any errors at all. I literally did nothing to my PC, nadda, it just started working all of a sudden!

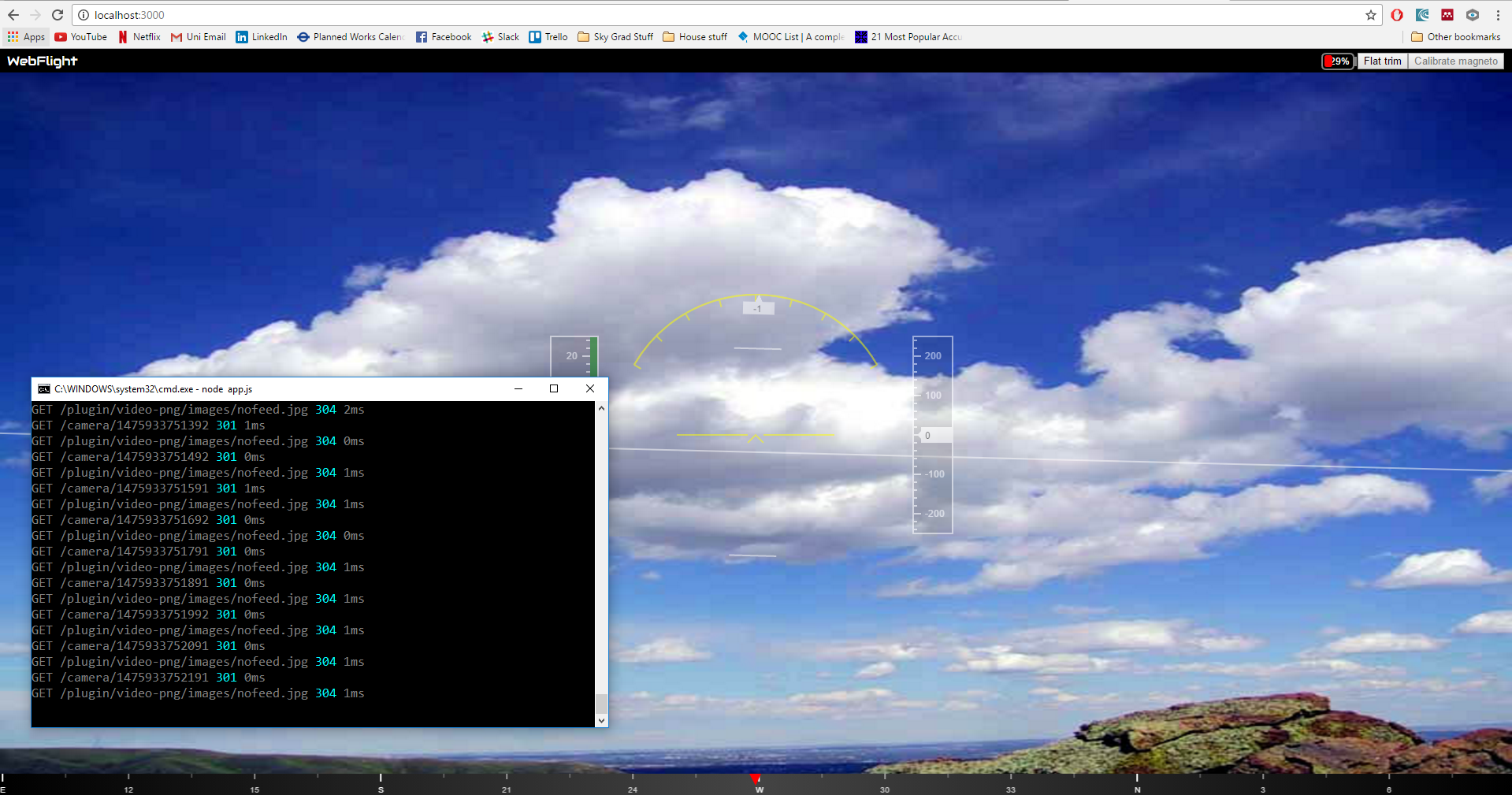
## 3. Configure Webflight

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 the 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), this will make sure that this file is 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 for controlling the drone. This file includes links to the config.js file (hence why renaming this a minute ago was important!) as well as bindings from the keyboard to the basic drone controls etc. There’s no need to edit this right now, that’s the basic configuration done!

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



You’ll notice that your drone is (probably) not pointed at a nice view of clouds, this is a stock image. The reason it displays this stock image is due to FFMPEG not being installed, this is the program used to render any images and videos from the drone to a format our browser can understand and output, so until we get that set up we’re stuck staring at the stock image!

Saying that though, it’s not all bad! We’ve got some nice HUD (Heads Up Display) elements including a compass along the bottom of the screen and some subtle instruments in the centre of the screen so we must be doing something right!

## 4. Download and configure FFMPEG

This section is a bit fiddly so take your time and go through it carefully!

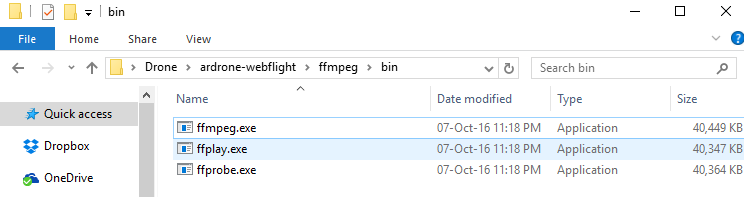
Downloaded FFMPEG (latest static 64-bit release) from here <https://ffmpeg.zeranoe.com/builds/>

In the previous guide I took you through how to find out if your PC/laptop is 32 or 64bit, the same rules apply to downloading FFMPEG. Find out whether your computer is using a 32-bit or 64-bit operating system as shown in section 2 of the previous guide <https://mveeprojects.wordpress.com/2016/10/15/flying-the-ar2-0-drone-beginners-guide-1/>).

When the download has finished, navigate to your Downloads folder and copy your new FFMPEG folder from there into your ardrone-webflight folder. As the folder is a zip file, we need to ‘extract’ it to use it. You can do this by right clicking it and selecting ‘extract here’. Rename the extracted folder as ‘ffmpeg’ for easier referencing later on.

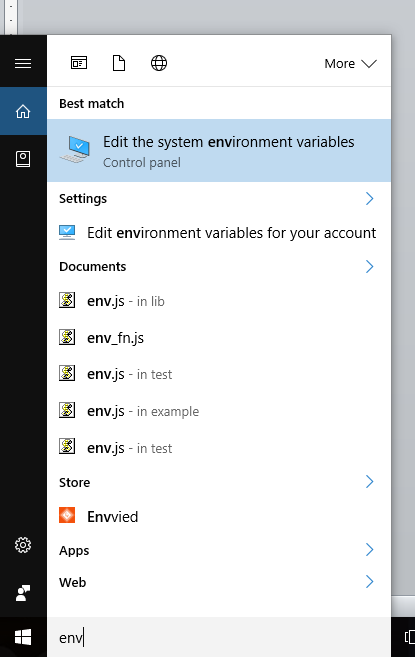
If you get stuck with installing FFMPEG have a look at this guide <http://www.wikihow.com/Install-FFmpeg-on-Windows>.

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.

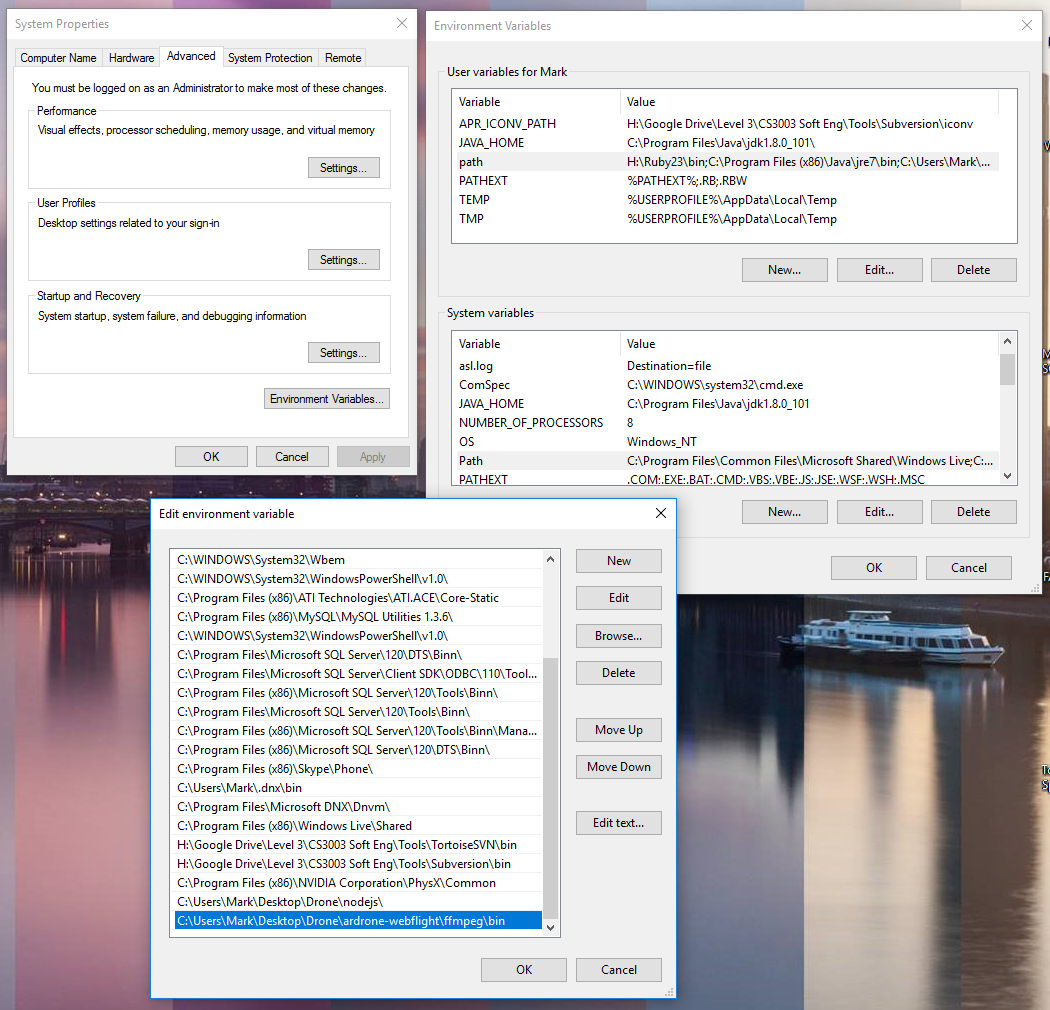


Now we need to edit the environment variables for your computer so you can use ffmpeg from anywhere. It would make sense that you could use the –g tag like we did earlier for bower, but I haven’t figured out why that doesn’t work in this case so I’m just putting it down to FFMPEG being awkward and doing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

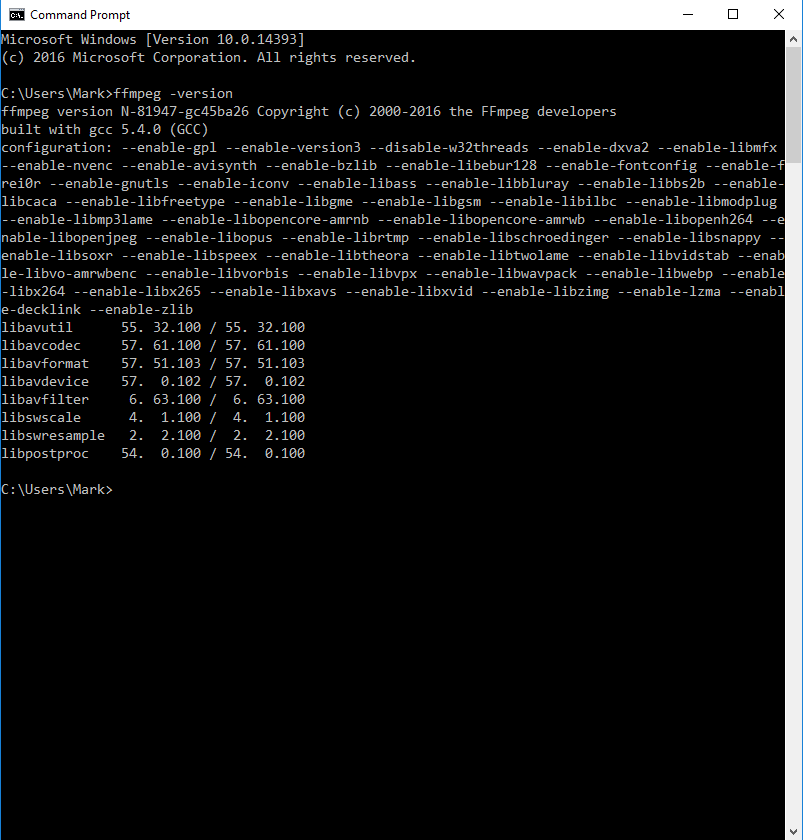
In Windows 10 all you need to do is start typing *‘env’* in the ‘Ask me anything’ search bar in the bottom left of your screen (Cortana I think it’s called?), you should see an option to edit the system environment variables as shown below, click this.



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.

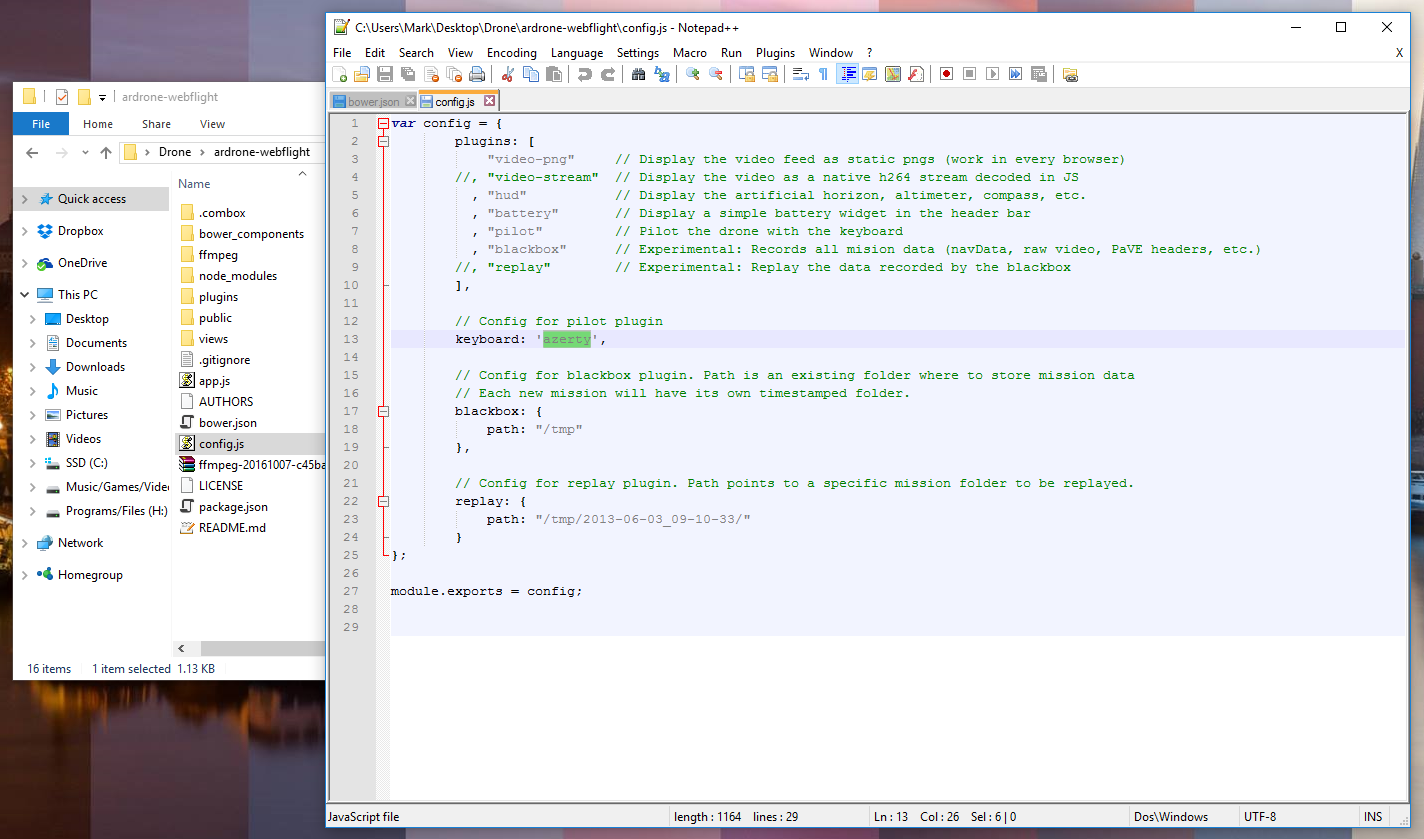


# 5. Controlling the drone using keyboard controls

Seeing the live feed from your drone’s camera 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, start Webflight up again with *‘node app.js’* (make sure you’ve closed down all other windows running Webflight first). You’ll notice that 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!)

# That’s all for now

With any luck you now have a drone flying around according to your controls, as well as a feed from the drones’ camera in the web browser (localhost:3000)! Check out Laurent Eschenauer’s Webflight repository on GitHub (<https://github.com/eschnou/ardrone-webflight>) for more controls, plugins etc. As always, if you ran into any issues getting up and running with Webflight please feel free to comment below and we’ll look into it together, alternatively head over to the WebFlight repository linked above and click the ‘issues’ tab to search any issues/fixes other users ran into!

In the next guide we’ll look into installing and some additional plugins for Webflight to extend its functionality even further!

Thanks for reading,

Mark