## Welcome

Welcome to my first guide on flying the Parrot AR 2.0 drone! These guides are based on my own experience of learning and working with the AR 2.0 drone so I’ll try to write these in a way that I would’ve found useful and easily understandable when I was getting started.

Although there are other great resources out there such as this Instructables guide (<http://www.instructables.com/id/Autonomous-AR-Parrot-Drone-20-Flying/step2/Node-JS/>) and this video (<https://www.youtube.com/watch?v=wQNdOuOxKSU>), which both achieve the same result, I wanted to write a guide for absolute beginners, explaining as much as possible as I go along. This way there is absolutely no expectation to have any prior understanding of programming or working with technology, I hope to assume as little knowledge as possible so please feel free to comment on this post if you have any questions or issues if you use this!

For the above reasons, this guide is a bit longer than most!

## Stuff you’ll need

Before we get in to anything, these are the things you’ll need to get going with this guide.

### 1. Windows 10 PC/laptop with Wi-Fi

For these tutorials, I am using a Windows 10 PC with a USB Wi-Fi adapter to connect to the drone, if you don’t already have Wi-Fi capability on your PC, something like the USB adapter in the link below should solve that for you.

<https://www.amazon.co.uk/TP-LINK-TL-WN725N-150Mbps-Wireless-N-Adapter/dp/B008IFXQFU/ref=sr_1_2?s=computers&ie=UTF8&qid=1475758246&sr=1-2&keywords=usb+wifi>

If you have a Windows 10 laptop, you should be all set (built-in Wi-Fi) ☺

**Windows:** Other versions of Windows should be fine, I initially completed some of this work on a Windows 7 laptop so that should be fine, Windows 8 (and 8.1) are notoriously awkward operating systems however so there may be a need to Google some bits and pieces if you run into issues. Feel free to comment on this post with any issues as well though!

**Linux/Mac:** At the moment I’m looking into getting a MacBook of my own so one day I should be able to try this all out on Mac and write a guide on that if needed. Might as well wait to do Linux until that point as well!

### 2. A drone!

Aside from that, you’ll also need a Parrot AR 2.0! You can find these on Amazon for under £200 brand new, but check the used/new options from other sellers as I managed to get it even cheaper through Amazon Warehouse Deals, just with a damaged box! The link below should give you an indication of the current cost of these drones from Amazon.

<https://www.amazon.co.uk/Parrot-Drone-Elite-Quadricopter-Sand/dp/B00FS7SSD6/ref=sr_1_3?ie=UTF8&qid=1476196262&sr=8-3&keywords=parrot+ar+drone+2.0>

### 3. Spare battery (optional, but recommended)

I would definitely recommend getting hold of an additional battery for the drone as the flight time on the standard battery (1000mAh Li-ion polymer) is around 10-12 minutes, give or take. I got hold of a 1500mAh battery to use while the standard one is charging (and vice-versa) which worked out quite nicely. The 1500mAh I bought is linked below and will allow for an absolutely breath-taking 18 minutes of flight time. This will cost around £40 which seems like a bit of a sting to the wallet but it with save you twiddling your thumbs for around 45 minutes while you’re waiting for a battery to charge before you can fly again!

<https://www.amazon.co.uk/Parrot-AR-Drone-1500mAh-Lithium-Polymer/dp/B00DAL5GD2/ref=sr_1_2?ie=UTF8&qid=1476196958&sr=8-2&keywords=parrot+ar+2.0+battery>

### 4. Notepad++

Aside from the above, I would recommend downloading Notepad++ as we’ll be fiddling with some code in JavaScript. Notepad++ is a nice, basic text editor that understands code formatting so will highlight elements of code for us, which will really help anyone get to grips with basic coding. You can download Notepad++ for free from the link below.

<https://notepad-plus-plus.org/download/>

## Intro

In this guide we are going to do the following:

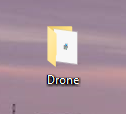
1. Install an application that will allow us to send and receive information to and from our drone (NodeJS)
2. Connect to the drone using Wi-Fi
3. Use NodeJS to tell the drone to take off and land from our laptop/PC
4. Write some simple JavaScript programs to save us typing lots of commands each time we want to fly the drone
5. Add some more functionality into our JavaScript programs and understanding the use of these to control the pitch, roll and yaw movements of the drone

To complete this, I have used a fantastic guide from the Instructables website (link below). This contains many of the commands that we’ll use, however I’ve tailored this guide to fill in any gaps I found in the Instructables guide and to make sure it works with the latest Windows operating system.

<http://www.instructables.com/id/Autonomous-AR-Parrot-Drone-20-Flying/step2/Node-JS/>

## Install NodeJs

Just before we get into installing NodeJS, the first thing I did was create a folder on the Desktop called ‘Drone’ which contained an empty folder called ‘nodejs’. When we install NodeJS below we’ll change the default install location from C:\Program Files\nodejs\ to C:\Users\**Mark**\Desktop\Drone\nodejs\ (just change ‘**Mark’** to the user name on your PC!)

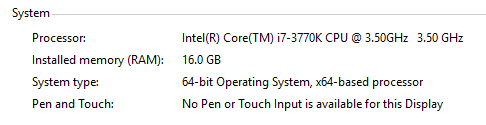


Now we’ve got a place sorted out for NodeJS to live, click the following link to view the available NodeJS downloads <https://nodejs.org/en/download/>.

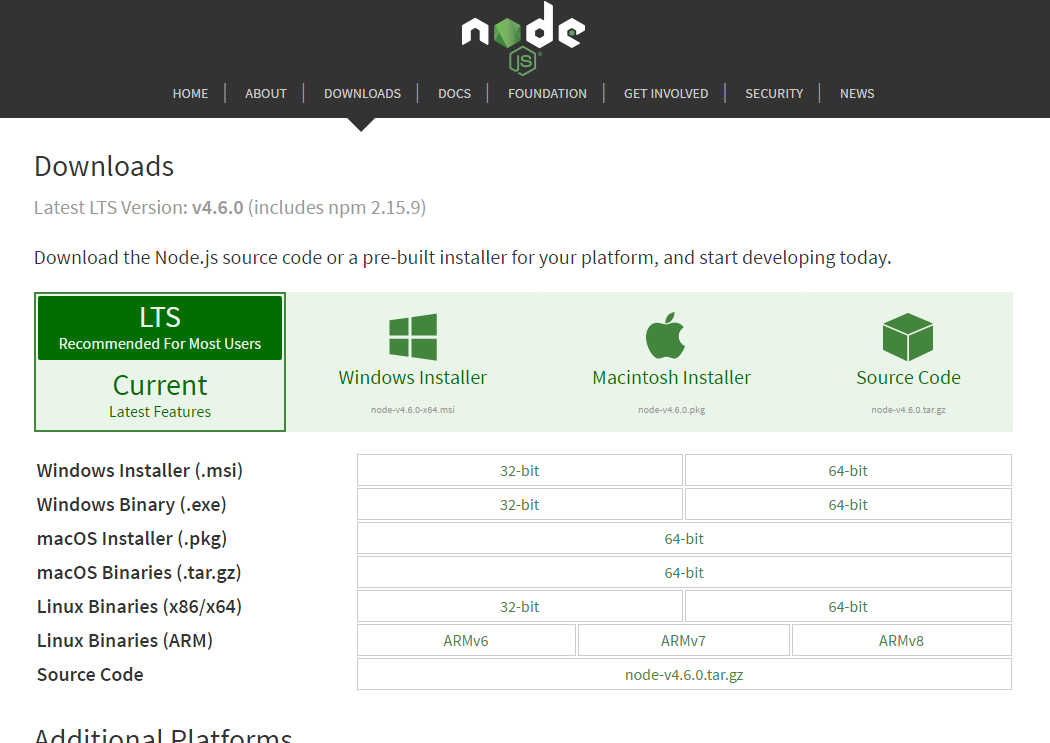
The version you download will ultimately depend on the operating system you are running, in Windows you can check this in the following ways:

* Windows 7 – right click the ‘Computer’ option in the start menu and click the ‘Properties’ option
* Windows 8 and 10 – hold down the windows key (key in-between CTRL and ALT on the bottom left of keyboard) and press E to open file explorer, then right click ‘This PC’ on the left hand side of the file explorer window and click ‘Properties’

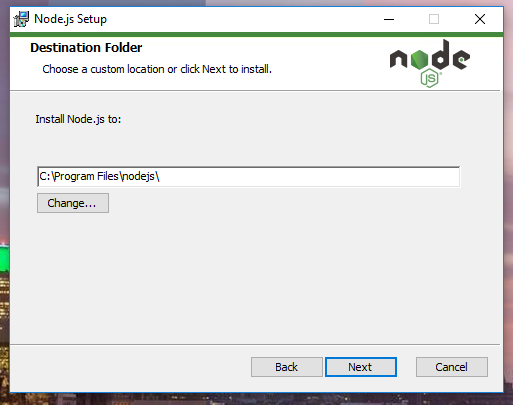
The resulting window should contain the ‘System’ section as shown below.



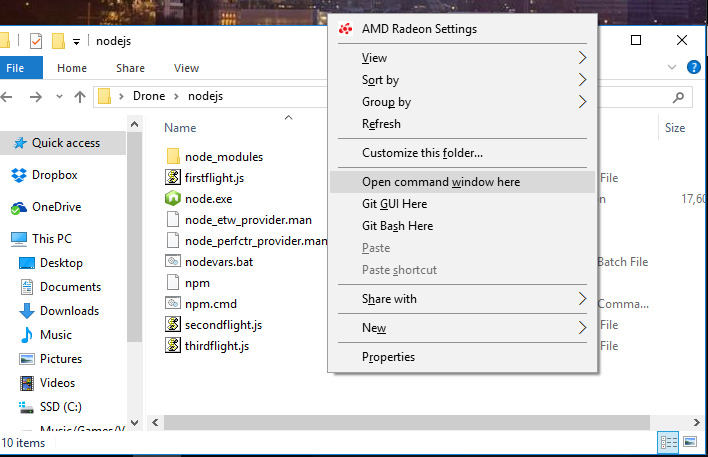
According to the result given above for my PC, I’m running a 64-bit operating system (‘System type’ field) so I’ll need to download the 64-bit version of NodeJS (use the .msi installer) as shown by the red box in the next screenshot.



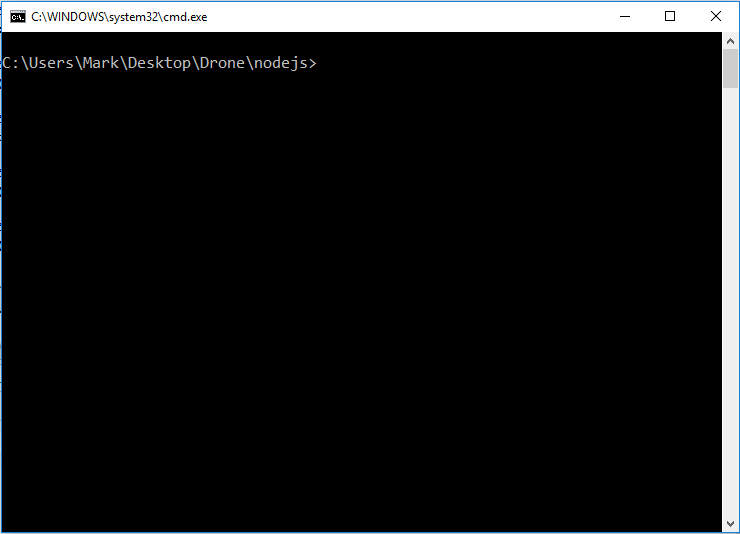
Clicking the link in the red box above will start the NodeJS download, once the file has downloaded; click it to start the installation process. Accept the license agreement and click Next on each screen if you’re happy with the defaults, apart from the screen that includes the below (install location of NodeJS). Click change and in the new window that opens, type ‘C:\Users\**<yoursystemusername>**\Desktop\Drone\nodejs\’ as the location to install NodeJS, click OK to confirm and close the new window and then click Next on the original window for all further steps of the installation.



We now have NodeJS installed on our machine! There’s just one last step to complete which will make sure that NodeJS knows what a drone is and how to work with it. To do this we need to use the windows command prompt within the nodejs folder – do this by holding down the left ‘shift’ button on your keyboard and right clicking and empty area inside the nodejs folder, then choose ‘Open command window here’ as shown below (ignore any additional files you can see in the background, you’ll create these soon!).

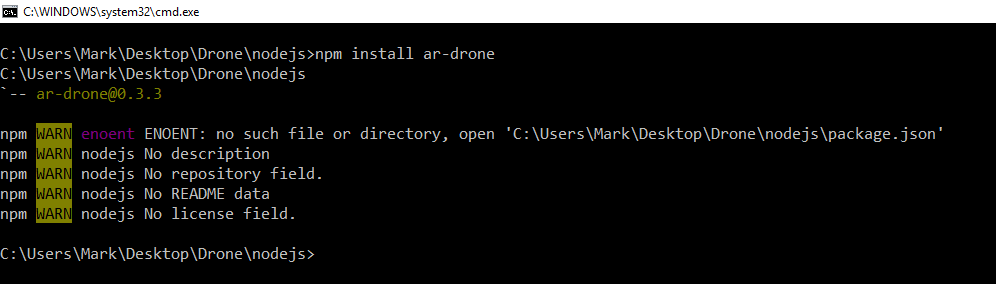


At this point you should have a command window open much like the one below.



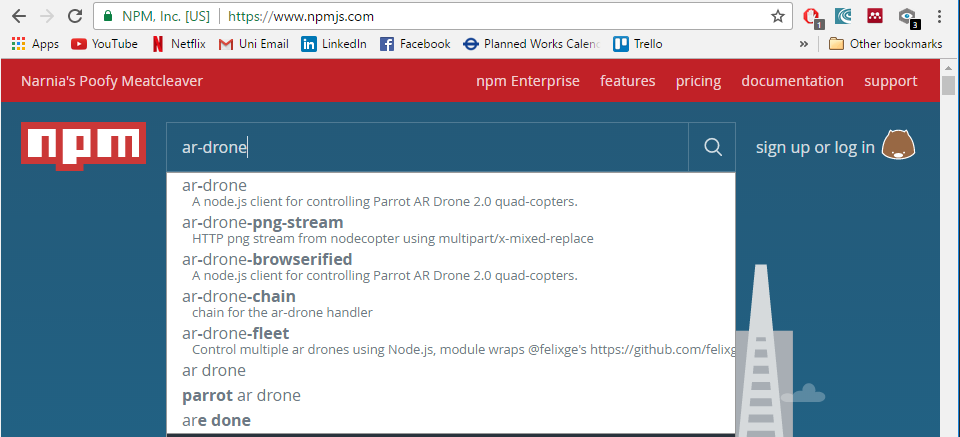
Type in the following and press enter; ***npm install ar-drone***

The result should look something like the output shown below, ignore the yellow warning (WARN) messages.



What this has done is tell the Node Package Manager (npm - picture this as a toolbox in a garage) that it needs a new ‘tool’, the tool in question is one created by Felixge which enables NodeJs to work with our ar-drone and perform commands such as take-off, land etc.

If you head to the npm website ([www.npmjs.com](http://www.npmjs.com)), you can search for available npm packages for a range of uses, below I’ve entered ‘ar-drone’ to see the packages available for our drone.

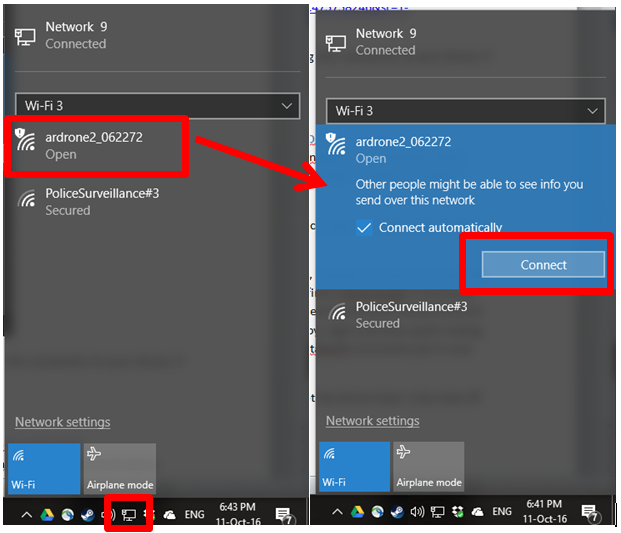


## Connect to Drone

First things’ first, if you’re using a desktop PC like me – make sure you have a wireless network adapter (see recommended USB Wi-Fi device mentioned earlier in this guide), and be sure to test it works by connecting to your Wi-Fi at home and accessing the internet through your web browser.

Attach a charged battery to the drone and set it on the side for now, the lights should come on and the propellers should perform a quick test (small jolt of each propeller, don’t worry it won’t just take off!).

Assuming your Wi-Fi is working as expected on your PC, a new connection should appear if you click the Network icon in the system tray (bottom right of your screen), ‘*ardrone2\_062272’* in my case. Click on your *ardrone2\_xxxxxx* network to open up the option to connect automatically (it sometimes won’t connect automatically, even if you tick this!) – either way, click the Connect button to create the connection.



**Note:** I’m not under police surveillance (promise), the Wi-Fi network we use was named by my housemate!

When you’ve connected to the drones’ Wi-Fi it will say ‘No internet, open’ regarding the connection to your drone, don’t worry about this at all, it’s only showing this as your PC would normally expect to a router/hub that provides internet access to your machine, our drone isn’t that clever unfortunately.

## Running NodeJS and manually controlling drone (takeoff and land)

Once you’re on the Wi-Fi of your drone, open the node program that we installed into ‘…Desktop\Drone\nodejs\’ earlier, double-click *node.exe* (green hexagon-shaped icon with a white ‘n’) to open the NodeJS terminal window.

Node will be your means of sending commands to the drone and receiving data back (such as receiving a live stream from the drone to your PC, which we’ll get into in the next guide). Now that we have the NodeJS terminal window open, we can start typing in some commands to explicitly establish our connection with the drone and start sending it commands.

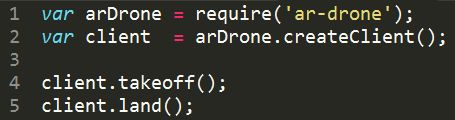
At this point I’m going to borrow some bits from the Instructables guide as they do an amazing job of going over the commands. To save you having to jump between different guides, I thought it would be better to copy and paste a couple of images from their guide (all credit for the below images of code to Instructables!)

In case you’d like to use the guide though, the Instructables guide can be found at the following link.

<http://www.instructables.com/id/Autonomous-AR-Parrot-Drone-20-Flying/step2/Node-JS/>

If you’re still with me, we’re going to start entering some stuff into the NodeJS command window we recently opened (node.exe). The image below shows four lines that each needs to be entered into the NodeJS command window, one-by-one. Before we enter these commands, let’s go over each one briefly:

* ***var arDrone = require(‘ar-drone’);*** this command lets Node know that it requires the library of tools called ***ar-drone*** to complete some or all of the commands that follow.
* ***var client = arDrone.createClient();*** we’re now getting a bit into networking, NodeJS is what’s called a ‘server-side’ language so its job is to provide stuff (commands, data, files etc.) to ‘clients’. In this case we’re telling Node that our drone is the client and we’re about to provide it some instructions, simple as that!
* ***client.takeoff();*** so we’ve told Node that our ‘client’ is the drone, this line is telling the drone to go ahead and run the command in the ***ar-drone*** library (see first command) called takeoff(). No prizes for guessing what happens when this command is executed!
* ***client.land();*** same idea as client.takeoff(); - it makes the drone land with the command called land() in the ***ar-drone*** library



So now that we’ve gone over each command hopefully you now have a better idea of how everything we’ve done up to this fits together. We need NodeJS to send data to and from our drone, but to do that we need the package/library of tools called *‘ar-drone’* to ensure NodeJS understands ***what*** a drone is and ***how*** to communicate with it.

Once you’re happy with the above ideas, go ahead and enter each command, I’ve written them out in text form below so you can copy and paste them into the Node window a line at a time if you like. Word of warning, when the drone takes off it will hover at about a meter off the ground and stay roughly there until you tell is to land. Just in case you need to stop/land it quickly for whatever reason, I would copy make sure you copy the command to land the drone (highlight command with cursor, right click, copy) before pressing enter on the take-off command! This will make sure you can paste the land command into the command window and press enter to stop the drone whenever you need.

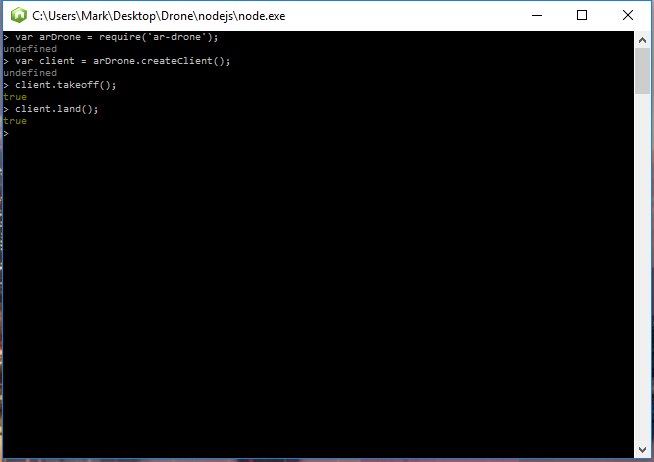
var arDrone = require('ar-drone');

var client = arDrone.createClient();

client.takeoff();

client.land();

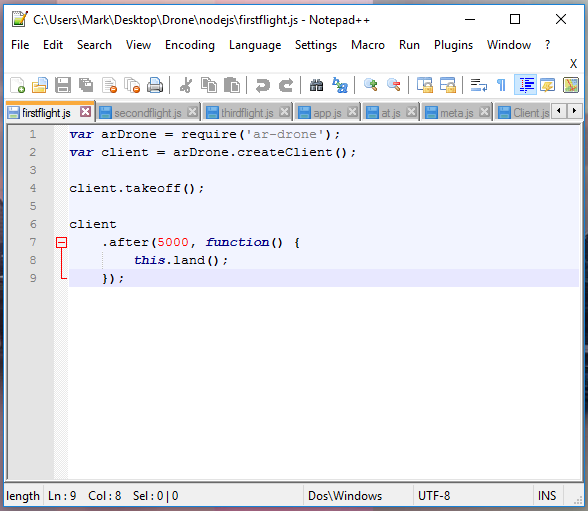
If you run into any errors, make sure each line ends with a semicolon (;) and you’ve run each line separately pressing enter after each semicolon. If it presents you with an error about not knowing what ar-drone is, make sure you’ve used the npm install ar-drone command as shown earlier in the guide. When all is said and done your drone window should look more or less exactly like mine below:



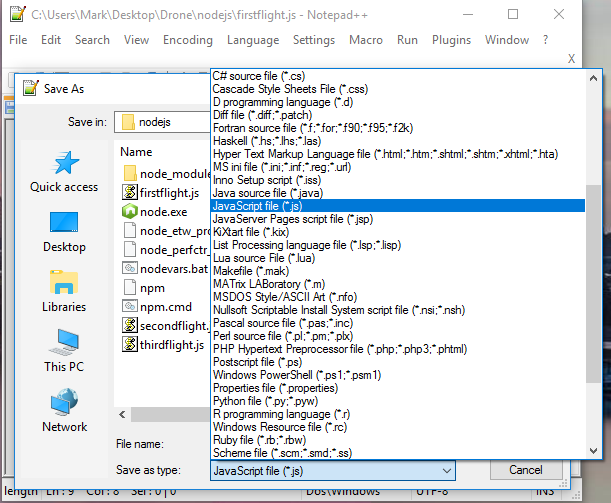
## Running commands as a simple, repeatable JavaScript program instead

For this I installed Notepad++ as this is a fantastic, free and simple text editor that comes equipped with code assistance tools which will help us more easily visualise the structure of our simple program. In case you skipped the link above, you can download Notepad++ here <https://notepad-plus-plus.org/download/>. You can just use Notepad it you like, Notepad++ is just a fancier version.

First thing’s first, let’s make a new notepad++ file for our program. Open Notepad++ from wherever you installed it and enter the code shown below:



Enter the program as shown below then save the file as *firstflight.js* in the same folder as your node.exe application (…\Drone\nodejs\firstflight.js). To do this click File -> Save As…, enter the File name field as firstflight.js and change the Save as type option to JavaScript file (\*.js) as shown below.



If you don’t fancy writing this file for yourself, firstflight.js and some other JavaScript files (which we’ll be going over soon) can be found on my GitHub repo here <https://github.com/MarkVee87/drone/tree/master/basicjsprogram>, feel free to download them and mess around with the code!

To run our firstflight.js program we’ll need to open the windows command prompt (not the NodeJs command window) in the location of your firstflight.js file and enter the following command. Remember, to open the command window in a location in your file explorer you can hold left-shift and right click the location then choose ‘Open command window here’.

***node firstflight.js***

Your drone should now take-off, and about 5 seconds after starting to take off it will start to land. So it’ll take-off, hover for a couple of seconds, then land.

**Note:** if your drone just sits on the ground with its lights flashing, it is likely you’ve got a NodeJS window open somewhere, make sure all NodeJS windows are closed! The above command calls node and tells it to run a particular program - n*ode firstflight.js* essentially means “computer, I want you to use a program called node [node.exe as we know it] to run a JavaScript program called firstflight.js, simple as that!

**Another note:** I asked you to open command prompt in the directory of the firstflight.js program you wrote, this isn’t mandatory, it just saves you having to type out the whole path to your program like ***node C:\Users\Mark\Desktop\Drone\nodejs\firstflight.js*** – that would be a pain to enter every time compared to ***node firstflight.js***!

## Adding more commands in JavaScript

We now have a simple program that makes the drone take-off, float about a bit, then land. Altogether not very exciting but we can add some more commands into the program to make it do a bit more for us!

An example of this is making it perform a simple route around an object. This is exactly what I did when I was learning about the use of NodeJS with the drone as shown in the video linked below.

<https://www.youtube.com/watch?v=eoMemf7xAfQ&feature=youtu.be>

Made a second JS file called secondflight.js (copied and renamed firstflight.js). Now going to fiddle with the following commands as described in the Instructables guide:

* up(speed) - has the drone gain altitude at a speed between 1 (max speed) and 0 (still).
* down(speed) - makes the drone reduce altitude
* clockwise(speed) - drone spins clockwise
* counterClockwise(speed) - drone spins counter-clockwise
* front(speed)/back(speed) - changes the pitch causing horizontal movement
* left(speed)/right(speed) - changes the roll causing horizontal movement
* stop() - keeps the drone hovering in place

The file secondflight.js demonstrates adding a bit more functionality to our drone program. In this file we’re telling the drone to take off, go forward a bit (front) at 20% (0.2) of maximum speed, then stop after 2 seconds of going forward (lines 10 and 11, .after(2000, function() {this.stop()}. The same is then repeated but instead of going forward again (front), we’re telling it to go backwards (back), before stopping and then landing.

That was pretty straightforward, so just to concrete the idea in our minds about how these commands are built and work with the drone, thirdflight.js demonstrates the concept of ‘yaw’ using the clockwise and counterClockwise commands. Yaw is a movement where the drone will stay at the same height and rotate in mid-air without going forward or backwards – like putting a pen flat on its side on a table and rotate it, this is yaw! See the image below for a visual of yaw in terms of an aeroplane, linear rotation on the perpendicular axis of the object.



Figure 1: http://machinedesign.com/site-files/machinedesign.com/files/uploads/2014/06/PRY.gif