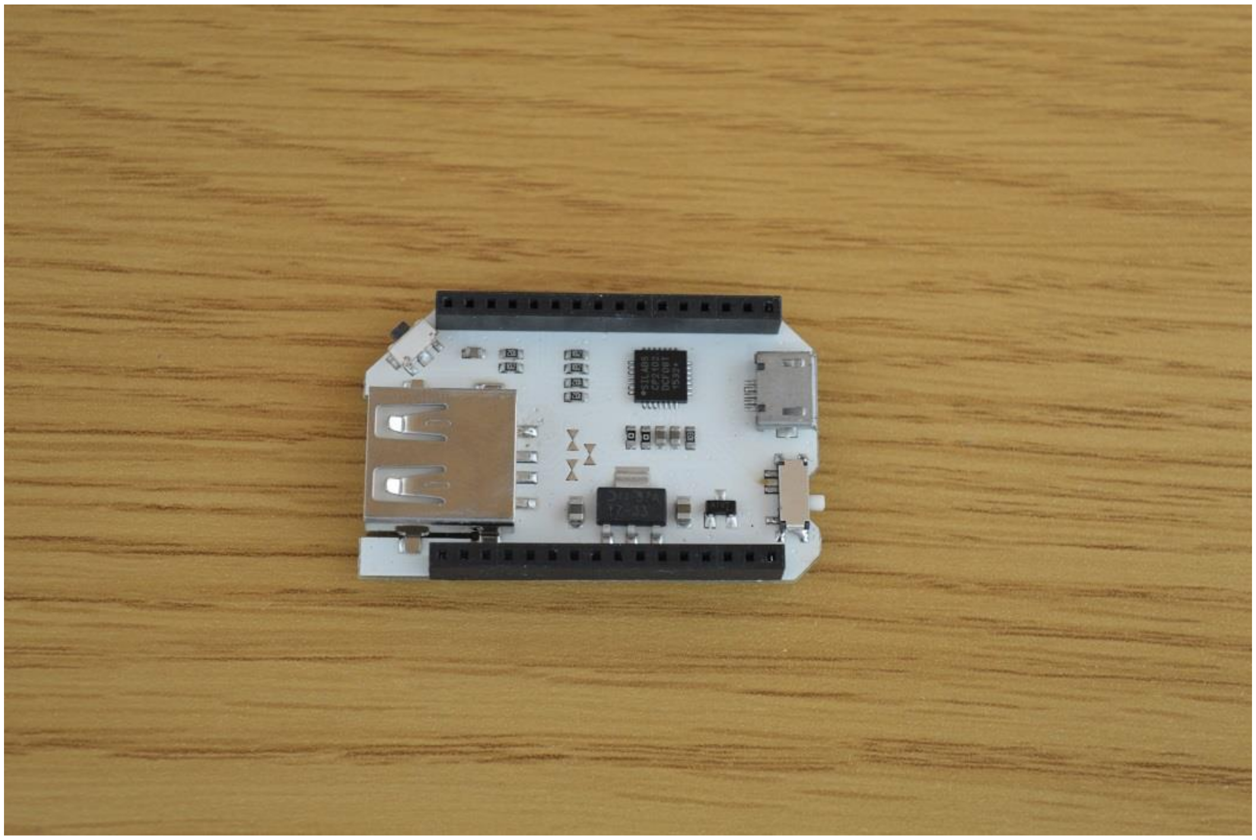
Mini Dock

The Mini Dock functions very similarly to the Expansion Dock. It supplies your Omega with power and allows you to communicate serially via a Micro-USB port. It also has a USB type A connector for you to use. All of this with a fraction of the size of an Expansion Dock.

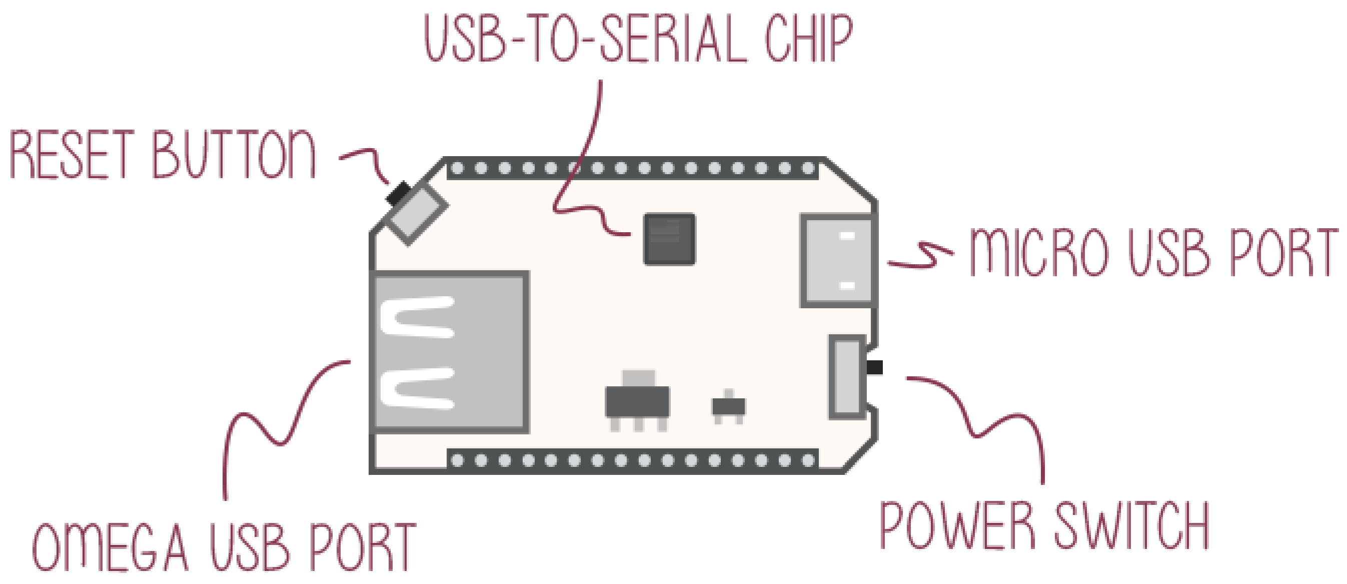


The Mini Dock is same size as the Omega. It does not have the Expansion Header of some of the other Docks, but this makes it perfect for Omega-only or USB-based projects.

The Hardware

The Mini Dock is tiny. It is approximately 4.3cm (1.7in) long, and 2.7cm (1.07in) wide.

The Mini Dock is powered by the Micro-USB port that supplies 5V to the Dock. This voltage is stepped down to the required 3.3V required to power the Omega, and also provides 5V to the USB Host port.

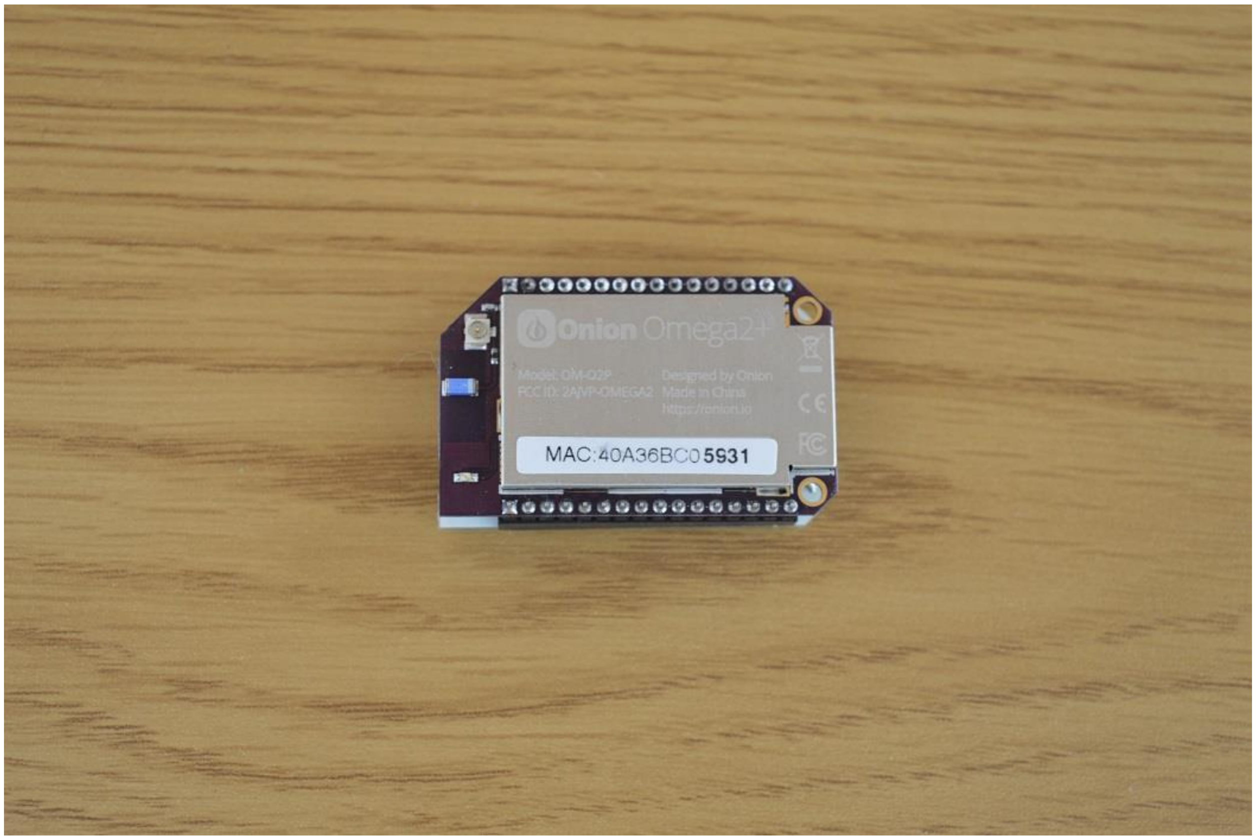


The Mini Dock allows for easy communication via the USB-to-Serial chip located in the center of the board.

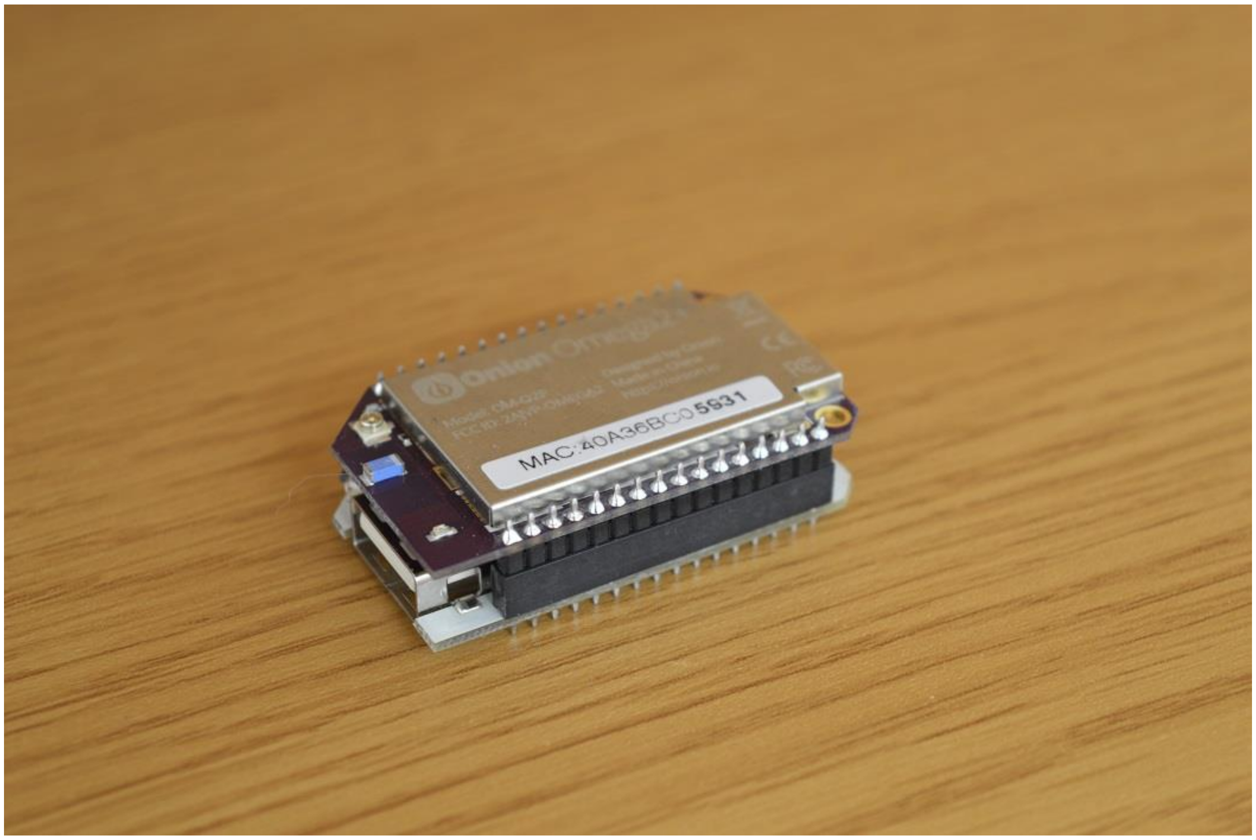
The reset button, located next to the USB Connector, can be used to quickly reboot your Omega, or you can hold it down for a factory reset if your Omega is ever in a bad state.

Connecting an Omega

To connect an Omega to the Mini Dock, line up the Omega’s edges with that of the Mini Dock’s as demonstrated below:



Make sure your Omega is pushed all the way down as demonstrated in the picture below:



You may need to line up the pins with the holes before pressing the Omega into the Dock.

The MicroUSB Port

The Micro-USB Port is used to supply power to the Dock, which in turn supplies power to the Omega.

The Micro-USB Port takes in 5V, and the Dock comes equipped with a voltage regulator to step the voltage down to 3.3V required for the Omega.

USB-to-Serial

The USB-to-Serial chip allows for a serial connection between the Omega and a computer using the Micro-USB port. You can connect a Micro-USB to USB cord from the Omega to your computer, open a terminal, and connect to the Omega via a COM port as opposed to SSH.

For more information on the Omega’s Serial connection read our guide to connecting to the Omega

Power Switch

The Power switch will cut power to the Omega, but not the serial chip. This means your computer will still detect a USB serial device, but will not be able to communicate with the Omega.

Reset button

The Reset Button on the Dock is connected directly to the Omega’s Reset GPIO. Pressing this button do one of two things: reboot, or factory restore.

Reboot

Momentarily pressing the reset button and letting go will initiate a reboot of the Omega OS.

Factory Restore

Pressing and **holding** the reset button for **10 seconds then releasing** will trigger a factory restore.

Warning: This will reset your Omega to the default filesystem of the last firmware update, **this will delete ALL of your data!**

Omega USB Port

The Omega’s USB Port can be used to connect to all sorts of devices, namely a USB storage device to extend the storage space of your Omega. The USB port supports USB 2.0, and is a type A connector.

Mechanical Drawings

We’ve made available a detailed diagram of the dimensions and geometry of the Mini Dock.

Connecting to the Omega’s Command Line

Now that your Omega is setup, connected to a WiFi network, and updated, you’ll want to connect to it to start building and inventing.

There are two ways to connect to the Omega’s command line:

- Using the local network to connect through SSH
- Using a USB connection to connect to the serial terminal

Both methods have their advantages and disadvantages. We recommend using SSH since it allows you to control wirelessly any Omega that's connected to your WiFi network.

The Command-Line Interface
The command-line is a way of interacting with a computer by sending commands in the form of single lines of text. This is different from "point and click" graphical user interfaces (GUI) found on most PC operating systems.

Command-line interfaces provide a more concise and powerful means to control a program or operating system, especially with regards to scripting (Shell Scripting, Python, etc).

This interface may seem overwhelming at first, but if you take the time to learn the basic commands you'll find that it's an incredibly powerful and useful tool to have in your toolbox.

Connecting with SSH

SSH stands for **Secure Shell** . It's a network protocol that creates a secure channel for communication between two devices on the same network. It can be used to secure many different types of communication, but here we'll be using it to login to the Omega's command prompt.

The Good & Bad of SSH

When using SSH, the Omega and your computer communicate over the WiFi network to which they are both connected. This means that as long as the Omega is powered on and within range of your WiFi network, you can connect to it! No need to plug it directly into your computer.

The disadvantage of SSH is that if the network connection gets interrupted, the connection will also be severed.

For most use-cases with the Omega, SSH will work really well. This should be your go-to method for accessing the Omega's command-line.

The Connection method

The connection method will be different depending on what Operating System you're using on the computer used to connect. We've included guides for the following:

- Mac OS X
- Linux
- Windows

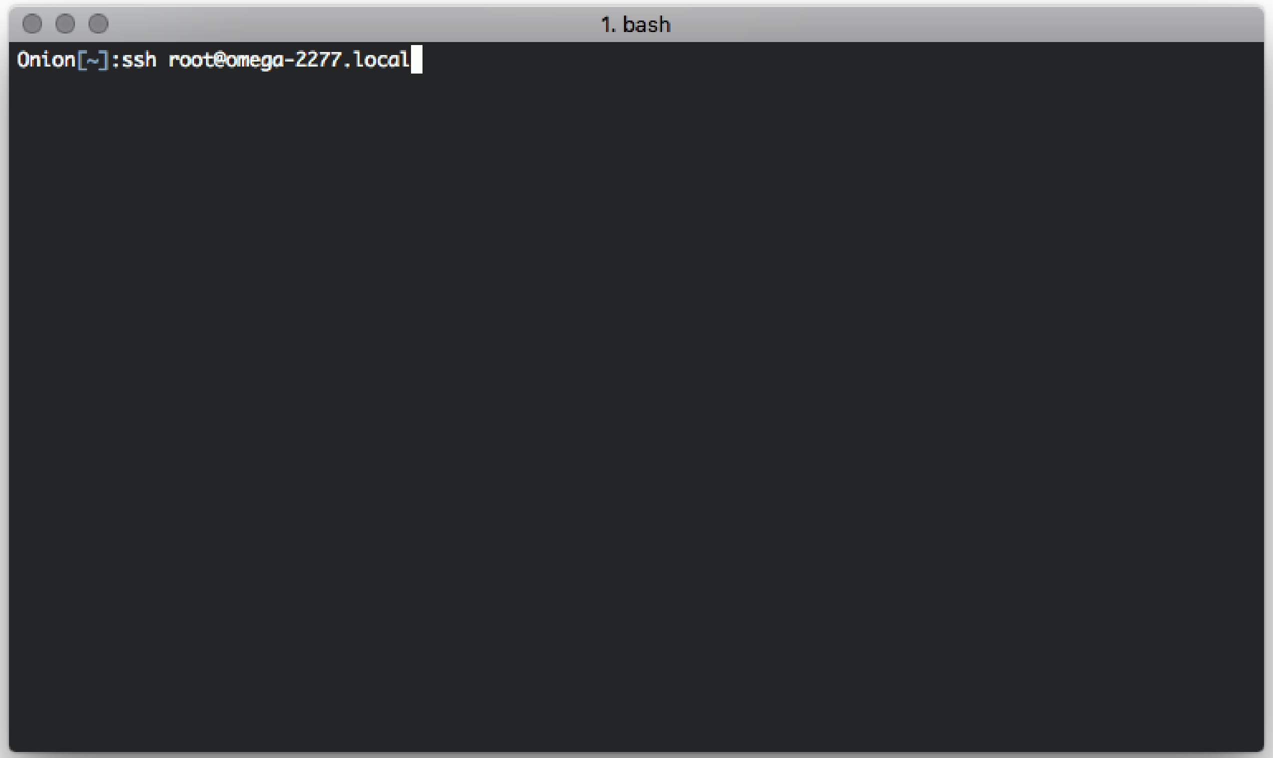
SSH on a Mac device

Step 1: Open a Terminal
Open your preferred Terminal App.

Step 2: Establishing a SSH connection
Run the following command:

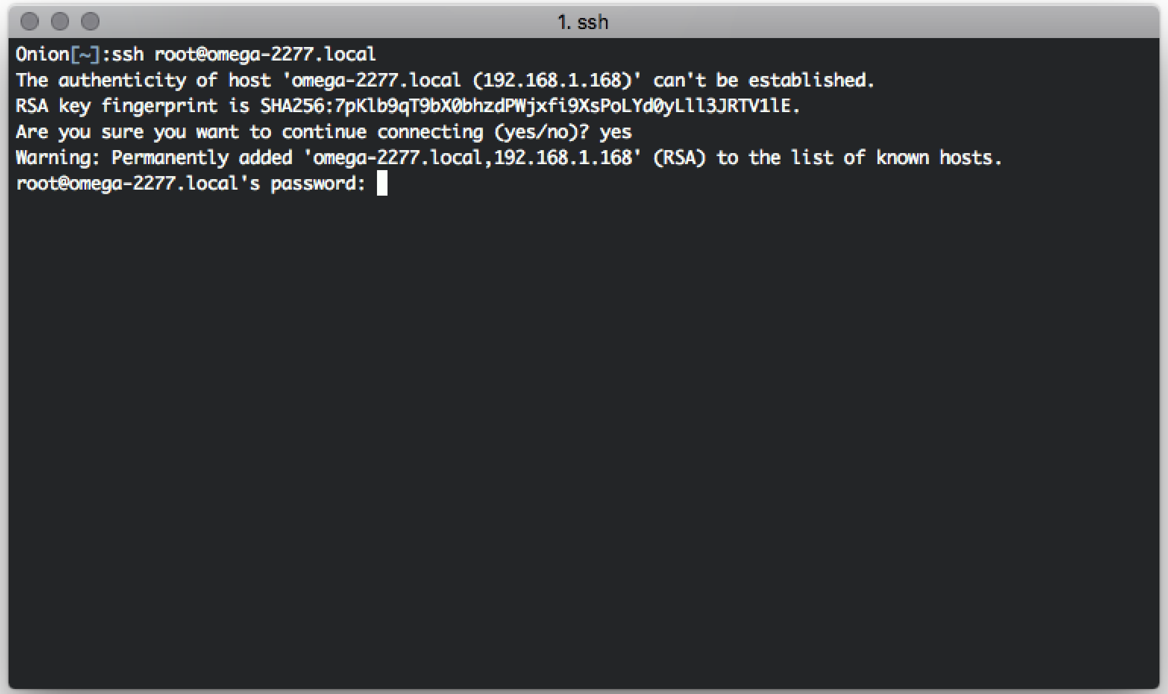
```
ssh root@omega-ABCD.local
```

Where `ABCD` is the unique id of your Omega.



```
Onion[~]:ssh root@omega-2277.local
```

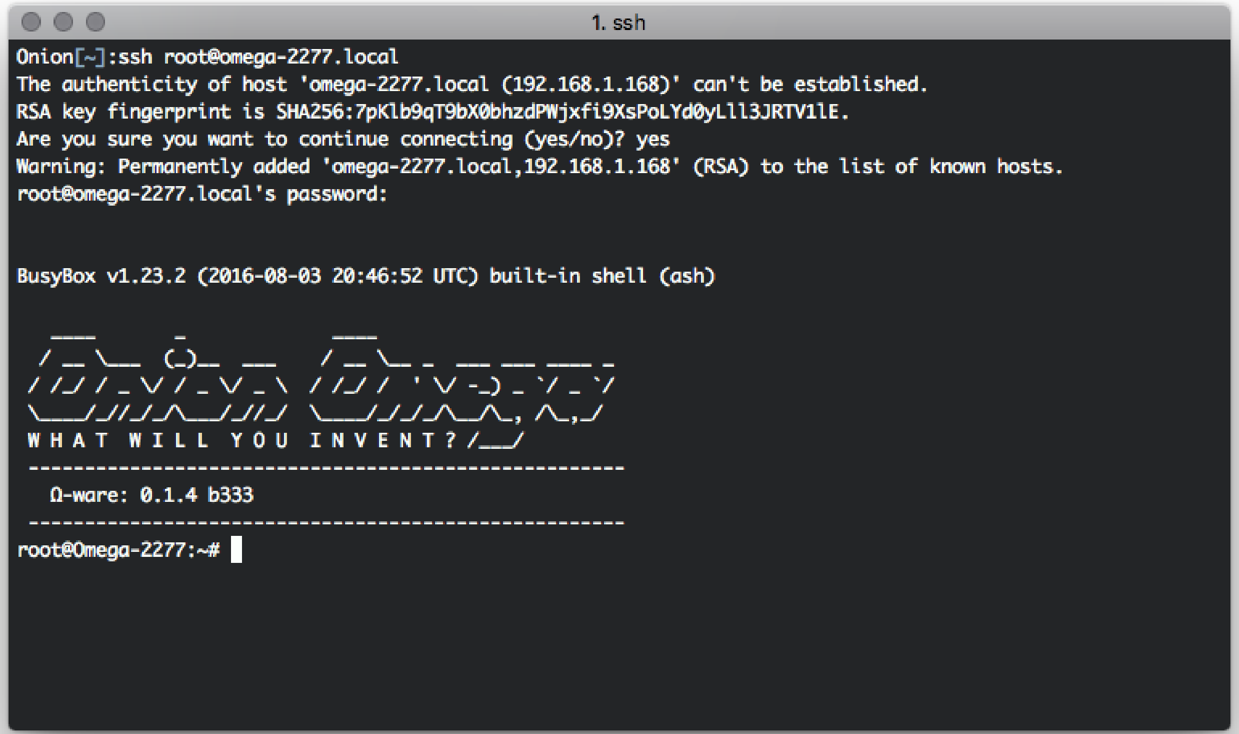
Step 3: Enter Credentials
When prompted, enter the password
By default, the password is: `onioneer`



```
Onion[~]:ssh root@omega-2277.local
The authenticity of host 'omega-2277.local (192.168.1.168)' can't be established.
RSA key fingerprint is SHA256:7pKlb9qT9bX0bhzdPWjxfi9XsPoLYd0yLll3JRTV1lE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'omega-2277.local,192.168.1.168' (RSA) to the list of known hosts.
root@omega-2277.local's password: 
```

If you're prompted about adding the address to the list of known hosts, type yes. This is just your computer getting to know the Omega for the first time!

And you're in!



```
Onion[~]:ssh root@omega-2277.local
The authenticity of host 'omega-2277.local (192.168.1.168)' can't be established.
RSA key fingerprint is SHA256:7pKlb9qT9bX0bhzdPWjxfi9XsPoLYd0yLll3JRTV1lE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'omega-2277.local,192.168.1.168' (RSA) to the list of known hosts.
root@omega-2277.local's password:


BusyBox v1.23.2 (2016-08-03 20:46:52 UTC) built-in shell (ash)

   ____       _             ____
  / __ \___  (_)__  ___    / __ \__ _  ___ ___ ____ _
 / /_/ / _ \/ / _ \/ _ \  / /_/ /  ' \/ -_) _ `/ _ `/
 \____/_//_/_/\___/_//_/  \____/_/_/_/\__/\_, /\_,_/
 W H A T  W I L L  Y O U  I N V E N T ? /___/
 -----------------------------------------------------
   Ω-ware: 0.1.4 b333
 -----------------------------------------------------
root@Omega-2277:~#
```

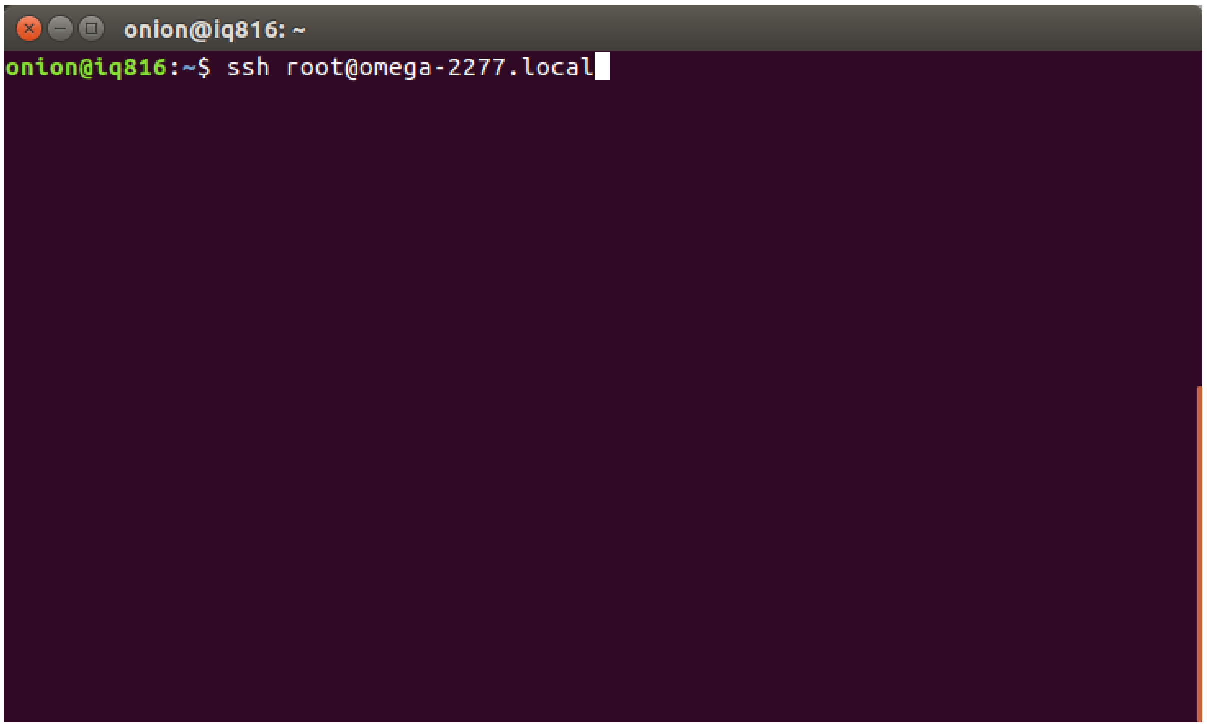
SSH on a Linux device

Step 1: Open a Terminal
Open your preferred Terminal App.

Step 2: Establishing a SSH connection
Run the following command:

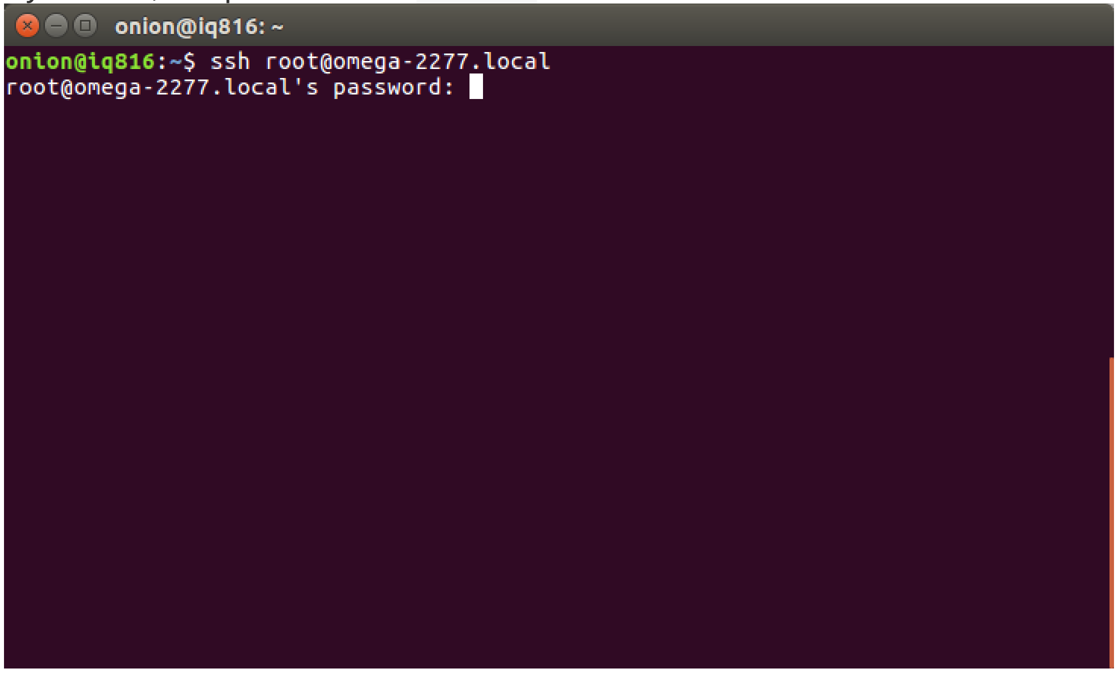
```
ssh root@omega-ABCD.local
```

Where `ABCD` is the unique id of your Omega.



```
onion@iq816:~$ ssh root@omega-2277.local
```

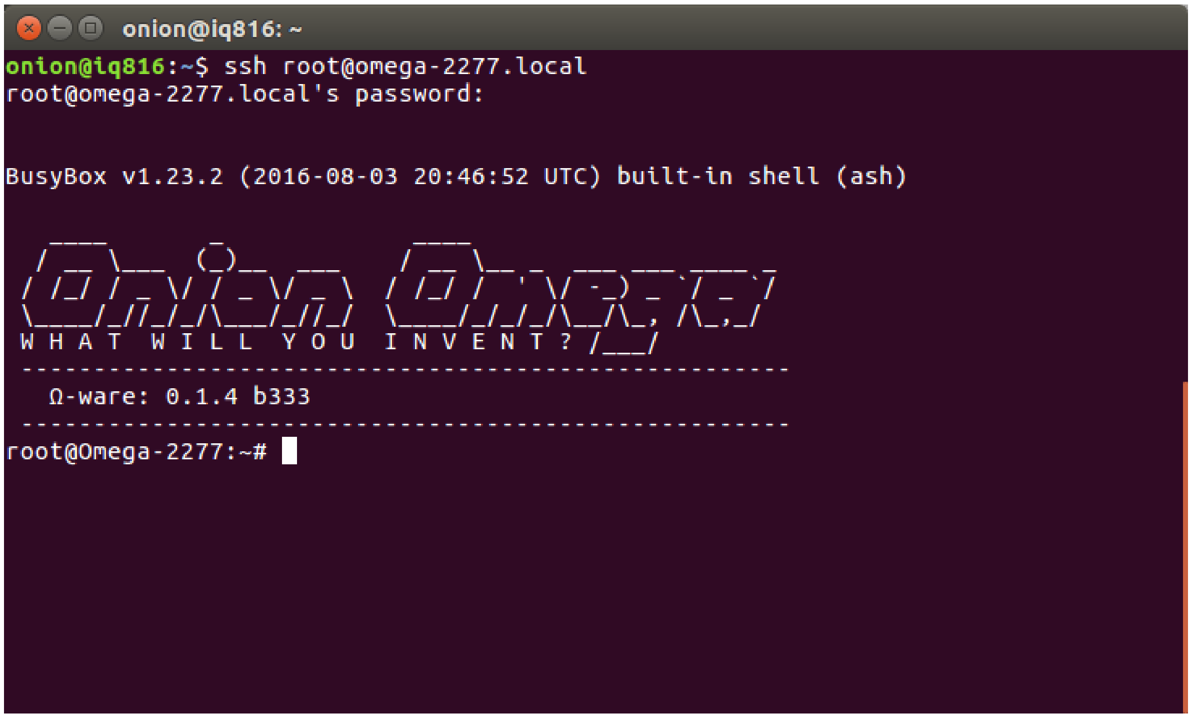
Step 3: Enter Credentials
When prompted, enter the password
By default, the password is: `onioneer`



```
onion@iq816:~$ ssh root@omega-2277.local
root@omega-2277.local's password:
```

If you're prompted about adding the address to the list of known hosts, type yes. This is just your computer getting to know the Omega for the first time!

And you're in!



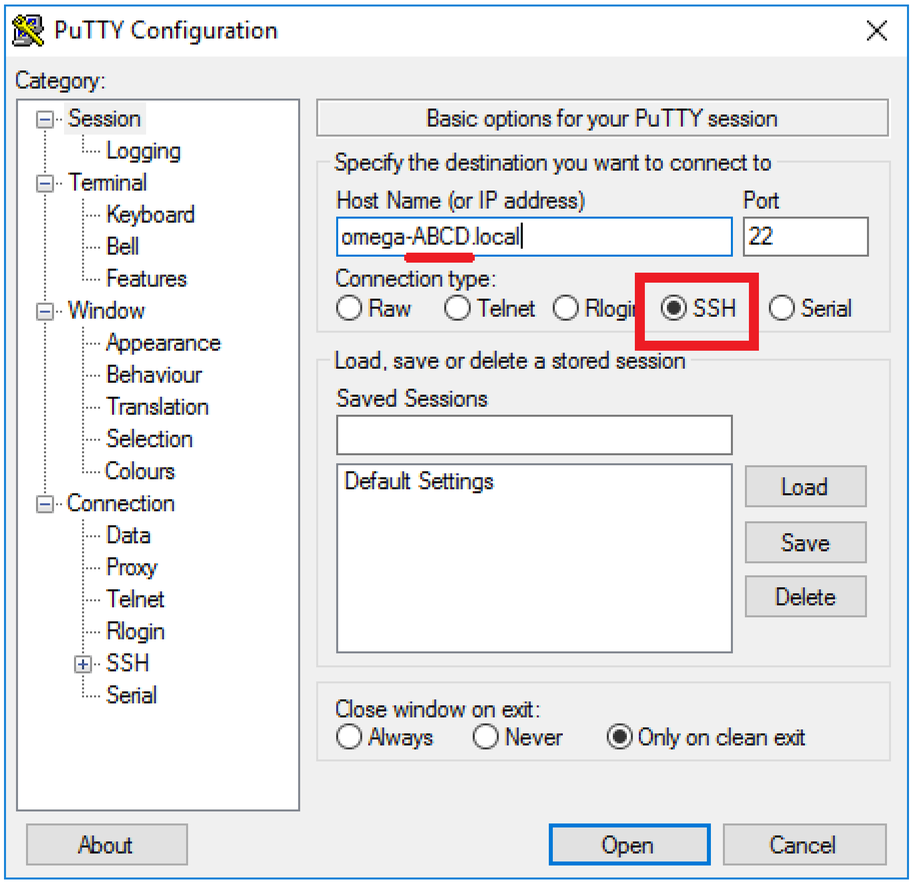
```
onion@iq816:~$ ssh root@omega-2277.local
root@omega-2277.local's password:


BusyBox v1.23.2 (2016-08-03 20:46:52 UTC) built-in shell (ash)

   ____       _             ____
  / __ \___  (_)__  ___    / __ \__ _  ___ ___ ____ _
 / /_/ / _ \/ / _ \/ _ \  / /_/ /  ' \/ -_) _ `/ _ `/
 \____/_//_/_/\___/_//_/  \____/_/_/_/\__/\_, /\_,_/
 W H A T  W I L L  Y O U  I N V E N T ? /___/
 -----------------------------------------------------
   Ω-ware: 0.1.4 b333
 -----------------------------------------------------
root@Omega-2277:~#
```

SSH on a Windows Device

Step 1: Download PuTTy
You can find PuTTy here. Look for the `putty.exe` installer for Windows on intel x86. Once it's downloaded you can open and use it.
Step 2: Establish a SSH connection
Configure an SSH connection to `omega-ABCD.local` on port `22`:



Specify the destination you want to connect to
Host Name (or IP address): omega-ABCD.local
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Close window on exit: Always Never [✓ Only on clean exit]

Where `ABCD` is the unique id of your Omega.
Step 3: Enter Credentials
Click Open and enter the credentials when prompted.

By default, the credentials are:
Username: `root`
Password: `onioneer`
And you're connected!

Using SSH Key Pairs

Over the course of a few months, the number of times you type in the password to connect will add up to a whole bunch of time that could have been spent having fun. Don't worry, there's another way to authenticate your SSH connection: **SSH Key Pairs**.

By using SSH Key Pairs, **the Omega and your computer will do a secure handshake so you won't need to type in that pesky password all the time**. Not only that, using a key pair will make your Omega even more since passwords can be discovered but secure key pair authentication cannot be broken.

You can skip this step if you want to dive right into using your Omega! You can always come back to it later :)

What are Key Pairs?

Good question! Authentication using a Key Pairs is based on having two randomly generated binary keys, where one is **public** and one is **private**.

- The private key is like a handwritten signature, used to prove your identity, so **make sure to keep it secret and keep it safe!**
- The public key's only purpose is to verify your identity, and is meant to be shared with other devices.

An SSH connection to your Omega that's secured by with a key pair will look something like this:

- Your computer will ask to login to the Omega, and the Omega will respond with 'Ok, but first prove your identity'
- Your computer will then generate a hash using your private key and send it to the Omega
- The Omega will use the stored public key to try to decode the identity hash, if the Public Key matches the Private Key, the decode will be successful, and the Omega will allow the connection to proceed.

Adding your Public Key

The method will be different depending on what Operating System you're using on the computer used to connect. We've included guides for the following:

- Mac OS X
- Linux
- Windows

How to Add your Public Key to the Omega on a MAC

Step 1: Locating Existing Key Pair

Let's first check to see if your computer already has a key pair. Open the Terminal App on your Mac and run:

```
ls ~/.ssh/id_rsa.pub
```

If this file exists, skip ahead to Step 3.

Step 2: Generating a Key Pair

No worries if you don't have a key yet, follow this quick guide to generate a key pair.

Step 3: Copy Key Pair

Copy the contents of the public key file to the clipboard:

```
cat ~/.ssh/id_rsa.pub
```

Step 4: Create an Authorized Keys File

Connect to your Omega’s command prompt and create a new file:

```
vi /etc/dropbear/authorized_keys
```

And paste your public key into it.

And you’re done!

From now on, you’ll be able to securely connect to your Omega without having to type out a password every time.

How to Add your Public Key to the Omega on a Linux machine

Step 1: Locating Existing Key Pair

Let’s first check to see if your computer already has a key pair. Open the Terminal App on your Mac and run:

```
ls ~/.ssh/id_rsa.pub
```

If this file exists, skip ahead to Step 3.

Step 2: Generating a Key Pair

No worries if you don’t have a key yet, follow this quick guide to generate a key pair.

Step 3: Copy Key Pair

Copy the contents of the public key file to the clipboard:

```
cat ~/.ssh/id_rsa.pub
```

Step 4: Create an Authorized Keys File

Connect to your Omega’s command prompt and create a new file:

```
vi /etc/dropbear/authorized_keys
```

And paste your public key into it.

And you’re done!

From now on, you’ll be able to securely connect to your Omega without having to type out a password every time!

How to Add your Public Key to the Omega on a Windows machine

Step 1: Locating Existing Key Pair

Let’s first check to see if your computer already has a key pair. Open the Windows Explorer and enter the following as the address:

```
%HOMEDRIVE%%HOMEPATH%\.ssh\id_rsa.pub
```

If this file exists, skip ahead to Step 3.

Step 2: Generating a Key Pair

No worries if you don’t have a key yet, follow this quick guide to generate a key pair.

Step 3: Copy Key Pair

Open the public key file with a text editor (Notepad works fine) and copy the contents to the clipboard

Step 4: Create an Authorized Keys File

Connect to your Omega’s command prompt and create a new file:

```
vi /etc/dropbear/authorized_keys
```

And paste your public key into it.

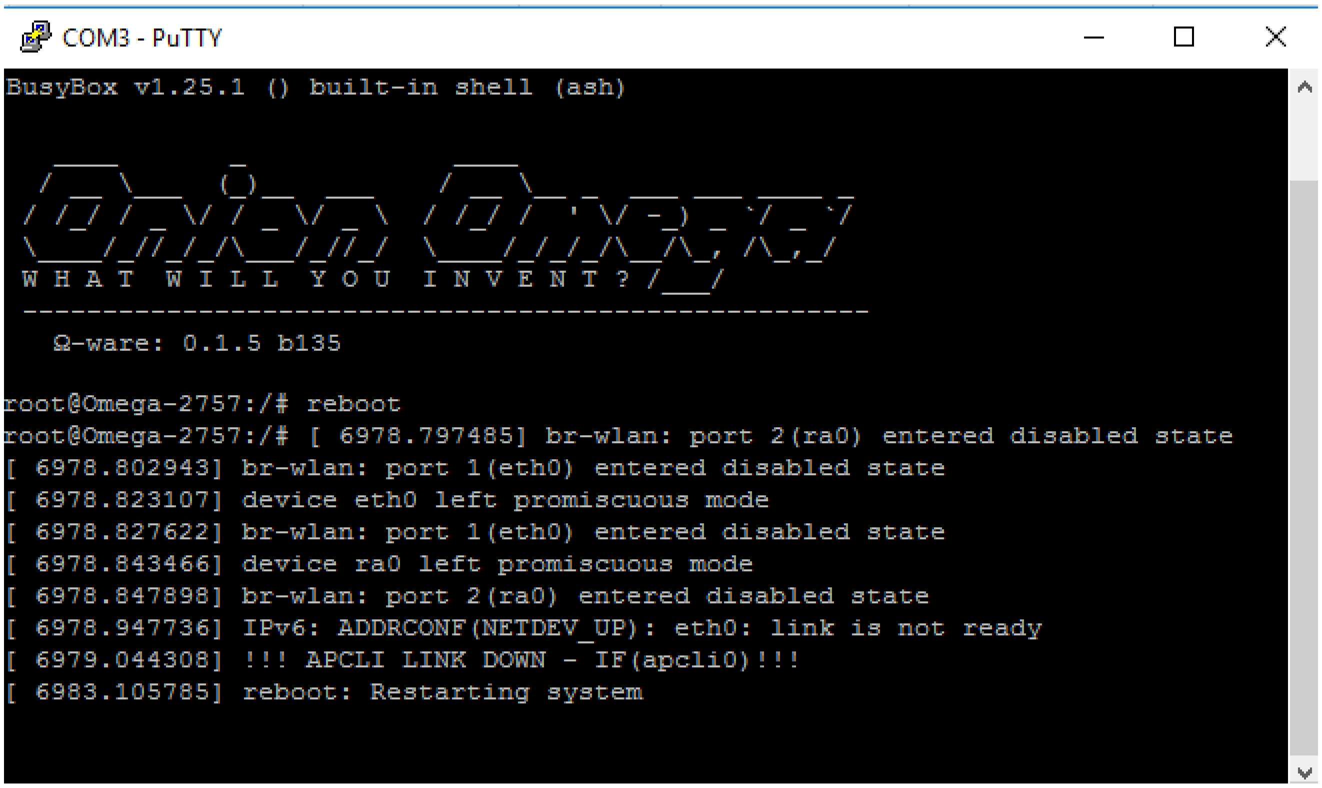
And you’re done!

From now on, you’ll be able to securely connect to your Omega without having to type out a password every time.

Connecting via Serial

The Omega’s command prompt can also be accessed with a USB cable, as long as your **Omega is docked in either an Expansion Dock or a Mini Dock**. What’s happening is that the Omega is using its UART pins to run a terminal, the USB-to-Serial chip found on the Dock is translating the Serial Terminal signals into USB signals that your computer can understand and vice versa.

Generally, we recommend using SSH to access the Omega’s command line, but the serial terminal does have its advantages. For instance, the serial terminal will always be available as long as the Omega is powered on and does not depend on network connectivity. Additionally, when using the serial terminal, you will see messages such as this one:



```
BusyBox v1.25.1 () built-in shell (ash)

   ____       _             ____
  / __ \____ (_)___  ____  / __ \____ ___  ___  ____ _____ _
 / / / / __ \/ / __ \/ __ \/ / / / __ `__ \/ _ \/ __ `/ __ `/
/ /_/ / / / / / /_/ / / / / /_/ / / / / / /  __/ /_/ / /_/ /
\____/_/ /_/_/\____/_/ /_/\____/_/ /_/ /_/\___/\__, /\__,_/
 W H A T  W I L L  Y O U  I N V E N T ? /____/
 -----------------------------------------------------
   Ω-ware: 0.1.5 b135

root@Omega-2757:/# reboot
root@Omega-2757:/# [ 6978.797485] br-wlan: port 2(ra0) entered disabled state
[ 6978.802943] br-wlan: port 1(eth0) entered disabled state
[ 6978.823107] device eth0 left promiscuous mode
[ 6978.827622] br-wlan: port 1(eth0) entered disabled state
[ 6978.843466] device ra0 left promiscuous mode
[ 6978.847898] br-wlan: port 2(ra0) entered disabled state
[ 6978.947736] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 6979.044308] !!! APCLI LINK DOWN - IF(apcli0)!!!
[ 6983.105785] reboot: Restarting system
```

This is an example of a message coming from the kernel. These messages can be listed out at any time using the `dmesg` command, so they can be seen when using SSH as well.
Note that the Expansion Dock and Mini Dock are the only docks that have USB-to-Serial chips, so the serial terminal will only work when using those docks. The serial terminal is meant for debugging during early development, for stable projects, SSH is the best method for accessing the command line.

We'll first identify the specific USB connection that we need to use to talk to the Omega, and then setting up the communication.

The Connection method

The connection method will be different depending on what Operating System you're using on the computer used to connect. We've included guides for the following:

- Mac OS X
- Linux
- Windows

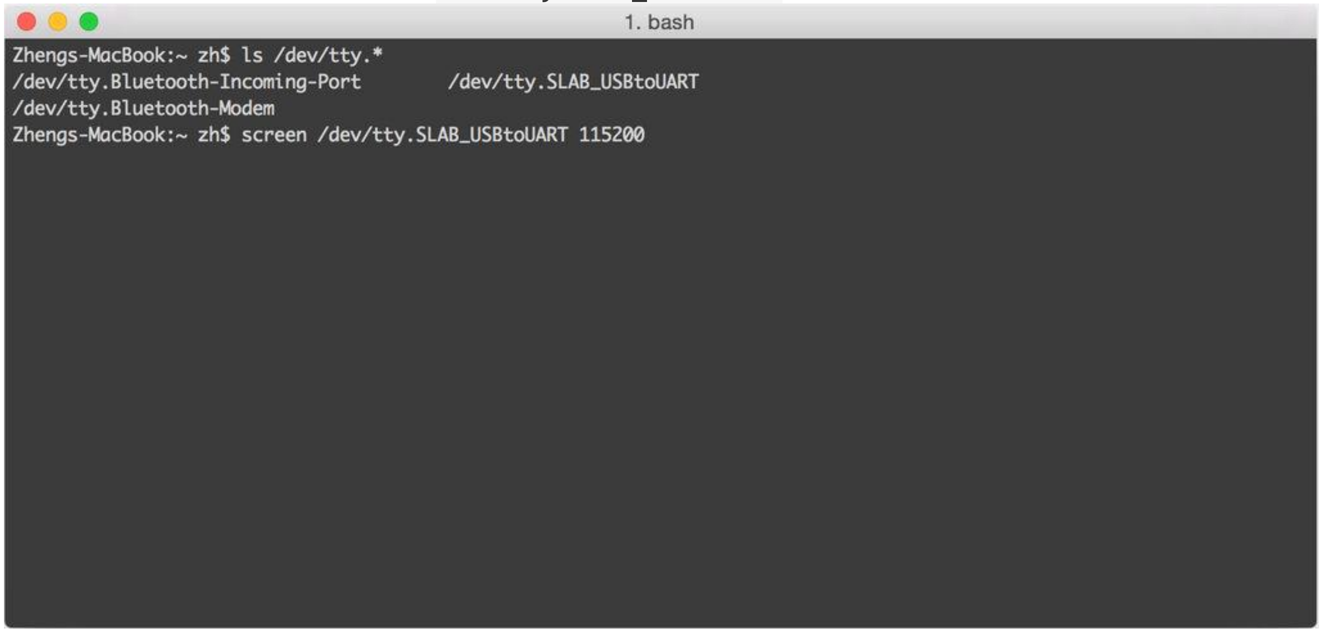
Serial on a Mac Device

Download Drivers

Download and install the Silicon Labs CP2102 driver for OS X.

Check if Serial Device Exists

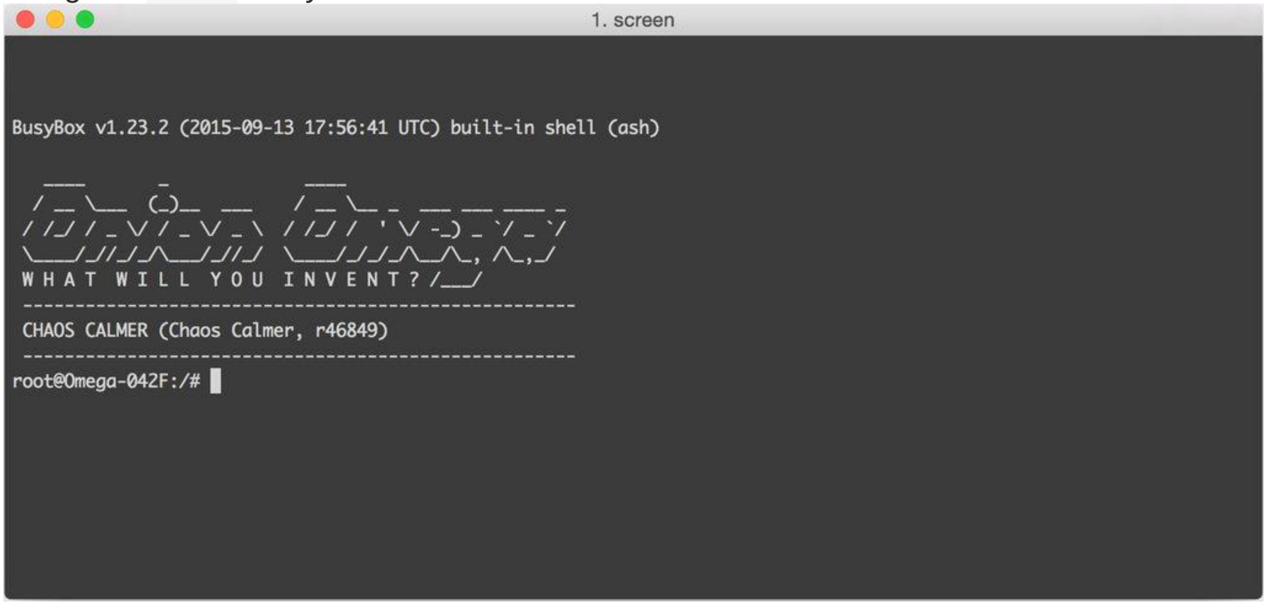
Plug in your Omega & Dock and run `ls /dev/tty.*` to see if the USB-to-Serial device can be detected. If the driver is successfully installed, you should be able to see a device with a name similar to `/dev/tty.SLAB_USBtoUART`.



```
Zhengs-MacBook:~ zh$ ls /dev/tty.*
/dev/tty.Bluetooth-Incoming-Port        /dev/tty.SLAB_USBtoUART
/dev/tty.Bluetooth-Modem
Zhengs-MacBook:~ zh$ screen /dev/tty.SLAB_USBtoUART 115200
```

Log in

Run `screen /dev/tty.SLAB_USBtoUART 115200` to connect to the Omega's serial terminal using the `screen` utility.



```
BusyBox v1.23.2 (2015-09-13 17:56:41 UTC) built-in shell (ash)

   ____       _             ____
  / __ \___  (_)__  ___    / __ \__ _  ___ ___ ____ _
 / /_/ / _ \/ / _ \/ _ \  / /_/ /  ' \/ -_) _ `/ _ `/
 \____/_//_/_/\___/_//_/  \____/_/_/_/\__/\_, /\_,_/
 W H A T  W I L L  Y O U  I N V E N T ? /___/
 -----------------------------------------------------
 CHAOS CALMER (Chaos Calmer, r46849)
 -----------------------------------------------------
root@Omega-042F:/# 
```

We recommend taking a peek at this tutorial to get an idea of how the `screen` utility works.

All Done

Enjoy! You're now connected to your Omega!

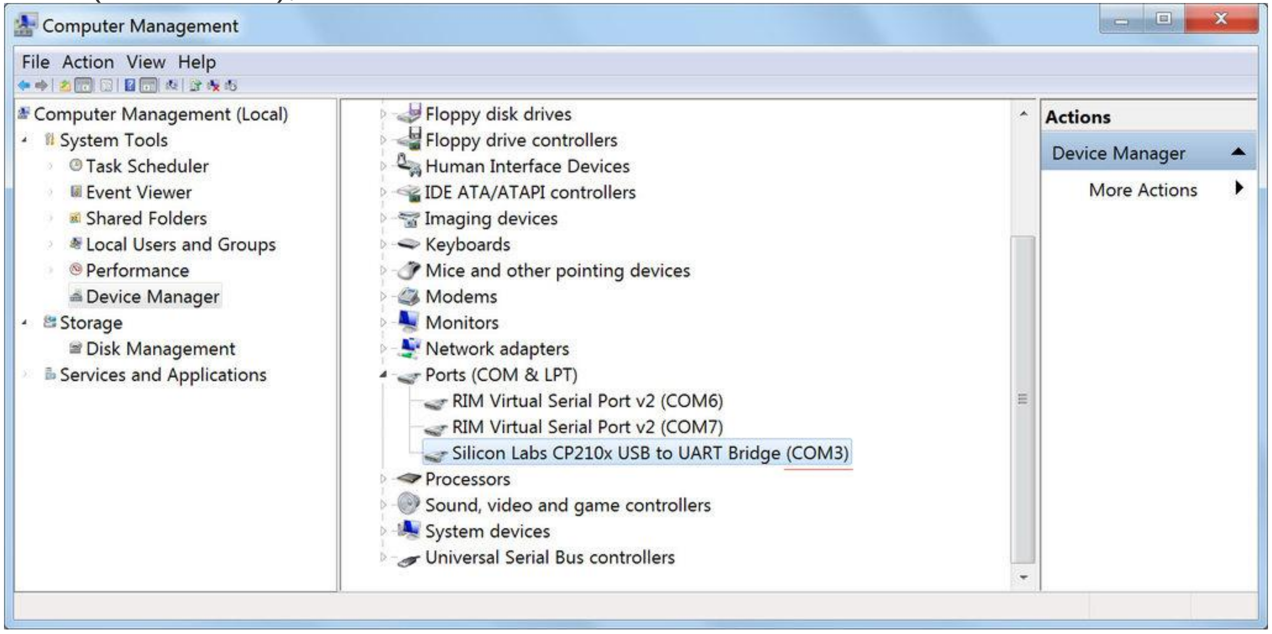
Serial on a Windows Device

Download Drivers

Download and install the Silicon Labs CP2102 driver for Windows.

Find Serial Device

Plug in your Omega & Dock and run Device Manager (Start > Enter "Device Manager" and press `ENTER`), look for Silicon Labs CP210x USB to UART Bridge under Ports (COM & LPT), and take note of the COM number in bracket.

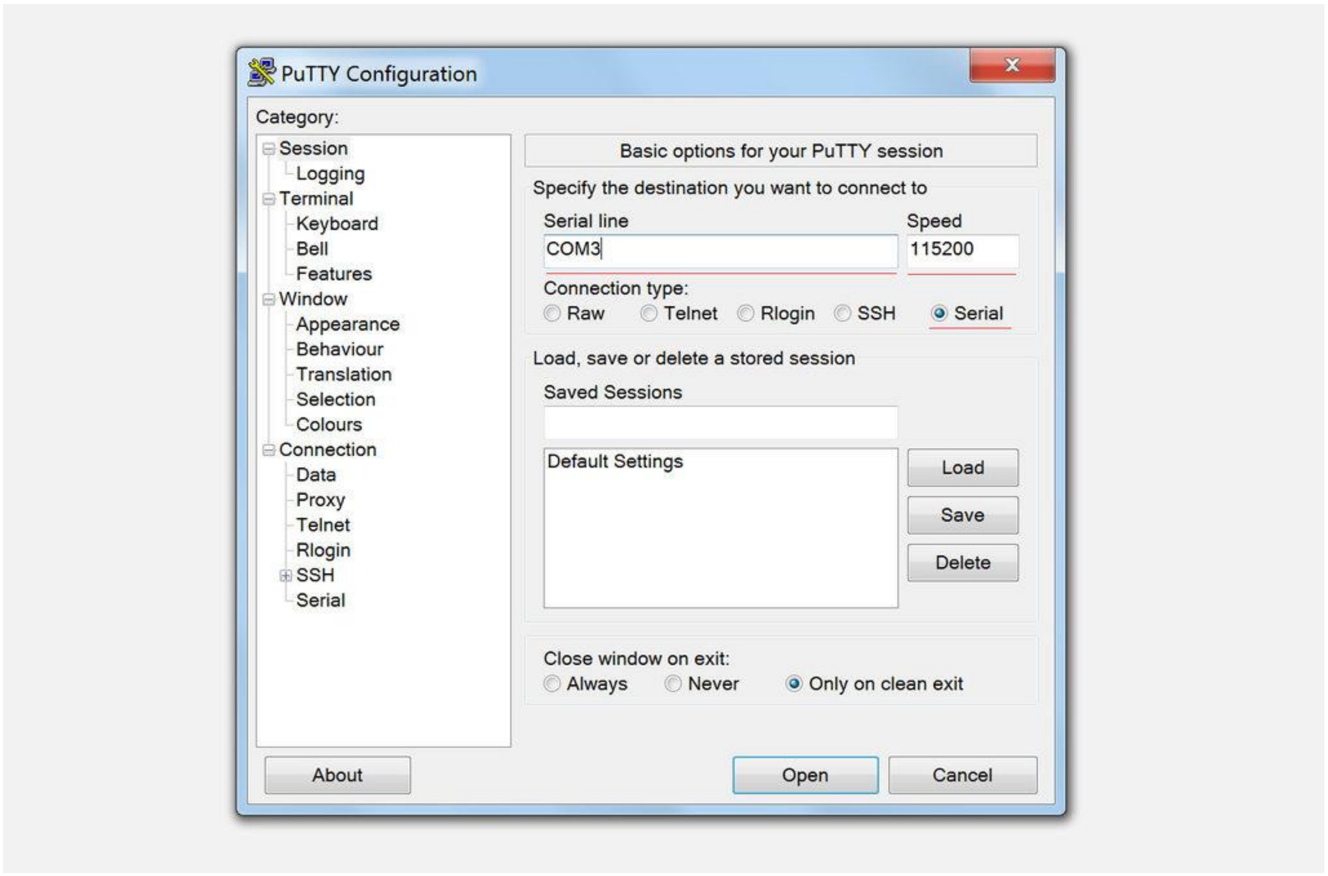


Download a Terminal Program

We'll be using PuTTy, but you can use another terminal program that you like. You can download Putty from this link.

Configure PuTTy

Open up PuTTy, select Serial for Connection type, enter the COM number noted down in Step 2 as Serial line, and enter `115200` for the speed.



Specify the destination you want to connect to

Serial line: COM3
Speed: 115200

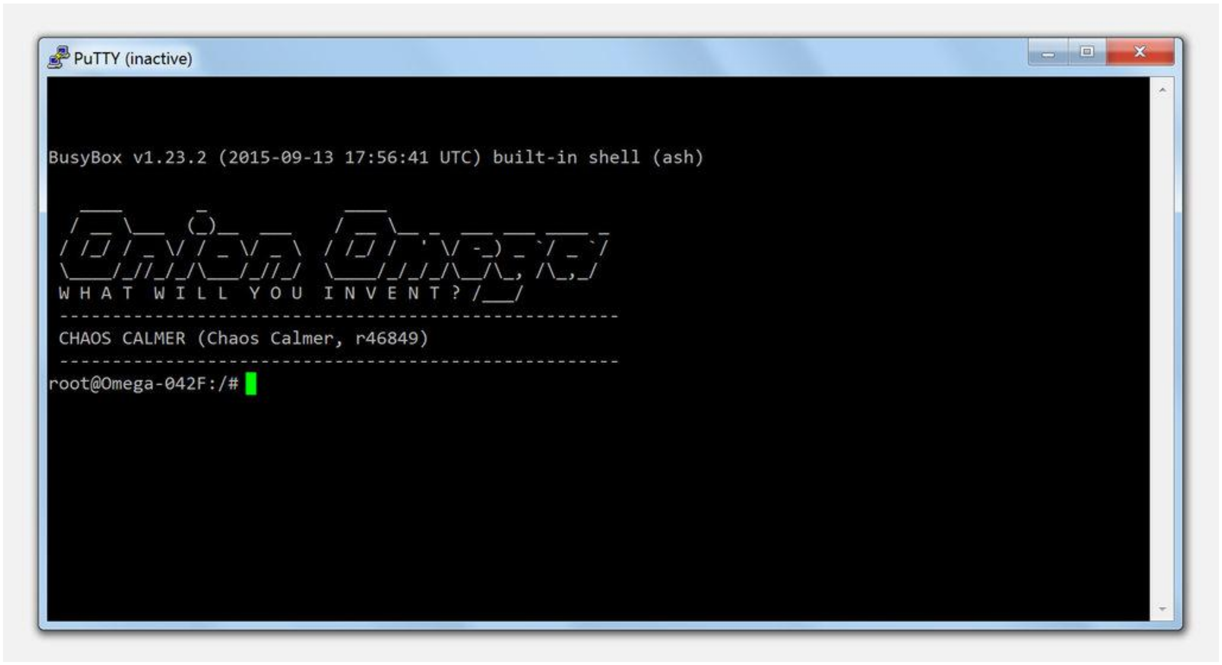
Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Saved Sessions:
Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

Connect

Click on the Open button to connect to the Omega via the serial terminal.



```


BusyBox v1.23.2 (2015-09-13 17:56:41 UTC) built-in shell (ash)

   ____       _             ____
  / __ \___  (_)__  ___    / __ \__ _  ___ ___ ____ _
 / /_/ / _ \/ / _ \/ _ \  / /_/ /  ' \/ -_) _ `/ _ `/
 \____/_//_/_/\___/_//_/  \____/_/_/_/\__/\_, /\_,_/
 W H A T  W I L L  Y O U  I N V E N T ? /___/
 -----------------------------------------------------
 CHAOS CALMER (Chaos Calmer, r46849)
 -----------------------------------------------------
root@Omega-042F:/#
```

All Done:

Enjoy! You're now connected to your Omega!

Serial on a Linux Device

Step 1: Check if the serial drivers are already installed

Some modern Linux Distros already have the required serial drivers installed.
Run `modinfo cp210x` on the command line, if it outputs several lines of information, the driver is already installed and you can skip ahead to **Step 4**.
If the output is something along the lines of

```
modinfo: ERROR: Module cp210x not found.
```

the driver will need to be installed. Continue to **Step 2**.

Step 2: Download and install the Silicon Labs CP2102 driver source files

For Linux kernel **3.x.x and higher**.

For Linux kernel **2.6.x**.

Step 3: Build and install the driver

For Ubuntu/Debian:

Unzip the archive.

`cd` into the unzipped directory.
Compile the driver with `make`.

```
sudo cp cp210x.ko /lib/modules/<kernel-version>/kernel/drivers/usb/serial/
sudo insmod /lib/modules/<kernel-version>/kernel/drivers/usb/serial/usbserial.ko
sudo insmod cp210x.ko
sudo chmod 666 /dev/ttyUSB0
sudo usermod -a -G dialout $USER
```

For RedHat/CentOS:

```
sudo yum update kernel* //need to update the kernel first otherwise your header
n't match
sudo yum install kernel-devel kernel-headers //get the devel and header packages
sudo reboot //your build link should be fixed after your system come back
```

Unzip the archive.

`cd` into the unzipped directory.
Compile the driver with `make`.

```
sudo cp cp210x.ko /lib/modules/<kernel-version>/kernel/drivers/usb/serial
sudo insmod /lib/modules/<kernel-version>/kernel/drivers/usb/serial/usbserial.ko
sudo insmod cp210x.ko
sudo chmod 666 /dev/ttyUSB0
sudo usermod -a -G dialout $USER
```

Step 4: Install `screen`

Let's install `screen`, a command line utility that will allow connecting to the Omega's serial terminal.
For Ubuntu/Debian:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install screen
```

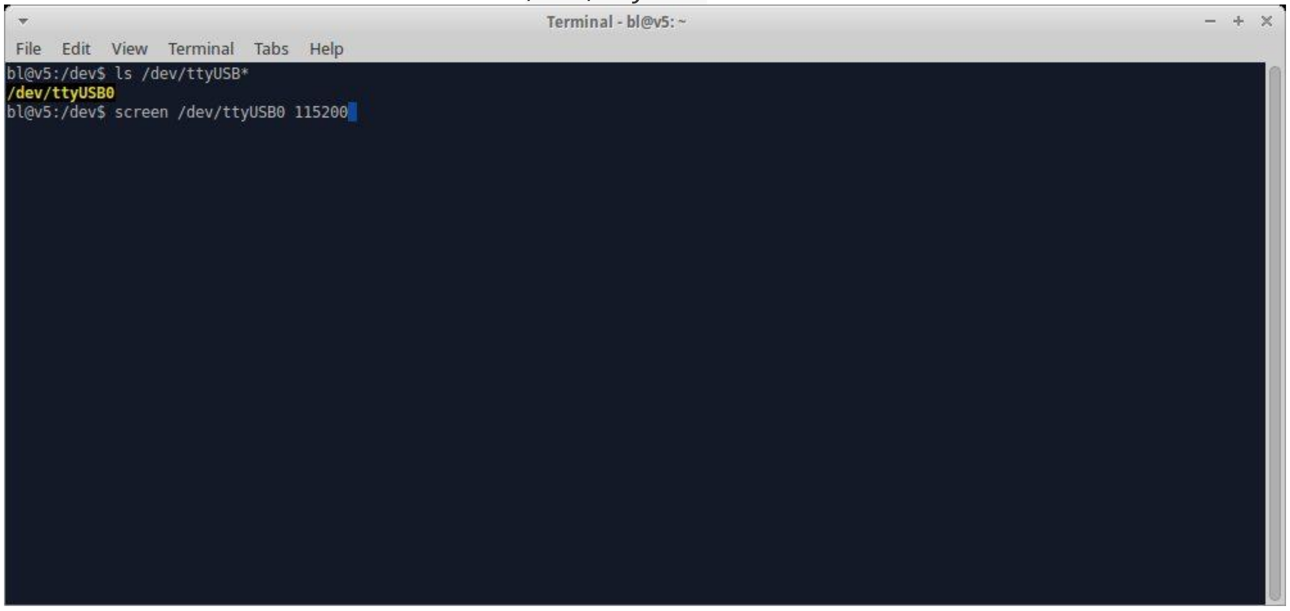
For RedHat/CentOS:

```
sudo yum update
sudo yum install screen
```

We recommend taking a peek at this tutorial to get an idea of how the `screen` utility works

Step 5: Look for the USB-to-Serial Device

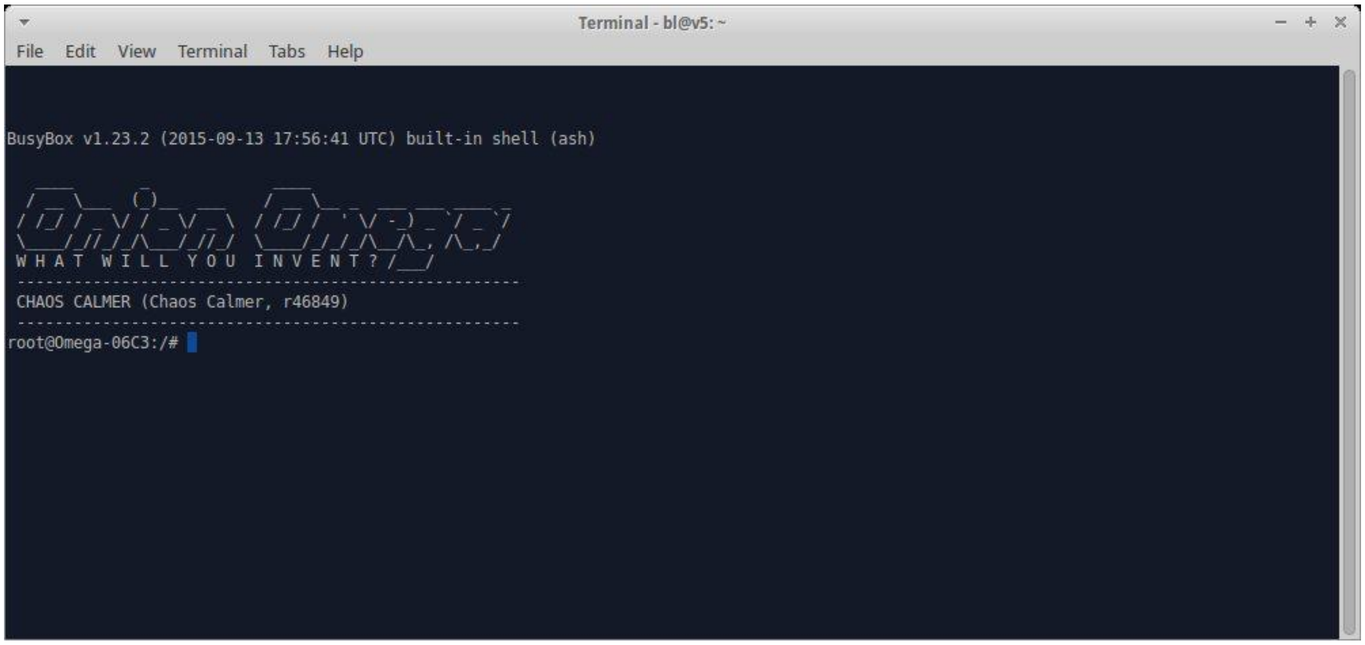
Plug in your Omega & Dock and run `ls /dev/ttyUSB*` to see if the USB-to-Serial device can be detected. If the driver is successfully installed, you should be able to see a device with a name similar to `/dev/ttyUSB0`.



```
bl@v5:/dev$ ls /dev/ttyUSB*
/dev/ttyUSB0
bl@v5:/dev$ screen /dev/ttyUSB0 115200
```

Step 6: Open Screen

Run `sudo screen /dev/ttyUSB0 115200` to connect to the Omega's serial terminal using screen.



```
BusyBox v1.23.2 (2015-09-13 17:56:41 UTC) built-in shell (ash)

   ____       _             ____
  / __ \___  (_)__  ___    / __ \__ _  ___ ___ ____ _
 / /_/ / _ \/ / _ \/ _ \  / /_/ /  ' \/ -_) _ `/ _ `/
 \____/_//_/_/\___/_//_/  \____/_/_/_/\__/\_, /\_,_/
 W H A T  W I L L  Y O U  I N V E N T ? /___/
 -----------------------------------------------------
 CHAOS CALMER (Chaos Calmer, r46849)
 -----------------------------------------------------
root@Omega-06C3:/#
```

If the screen remains blank, hit enter again to get to the command prompt.

Step 7:

Enjoy! You are now connected to your Omega!