

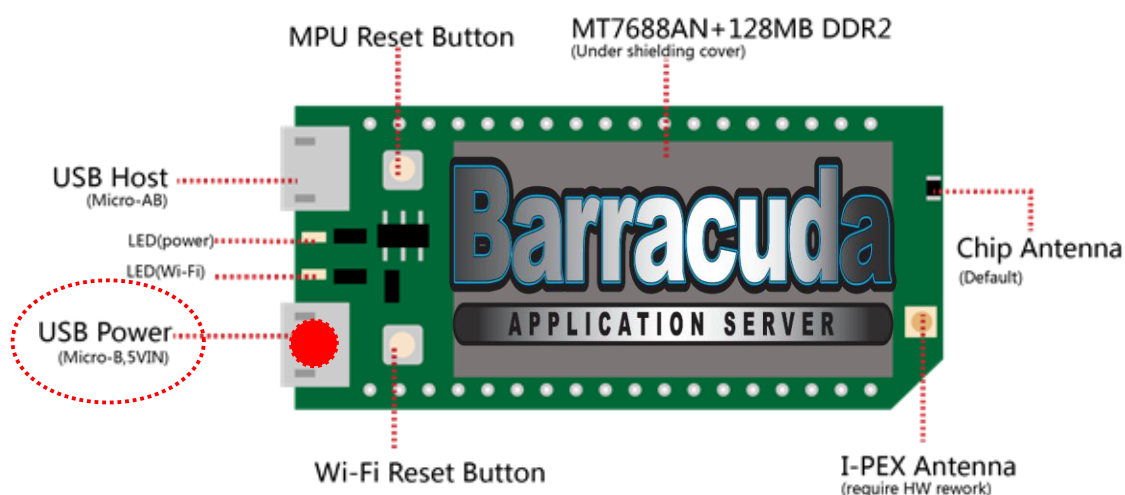
Barracuda

APPLICATION SERVER

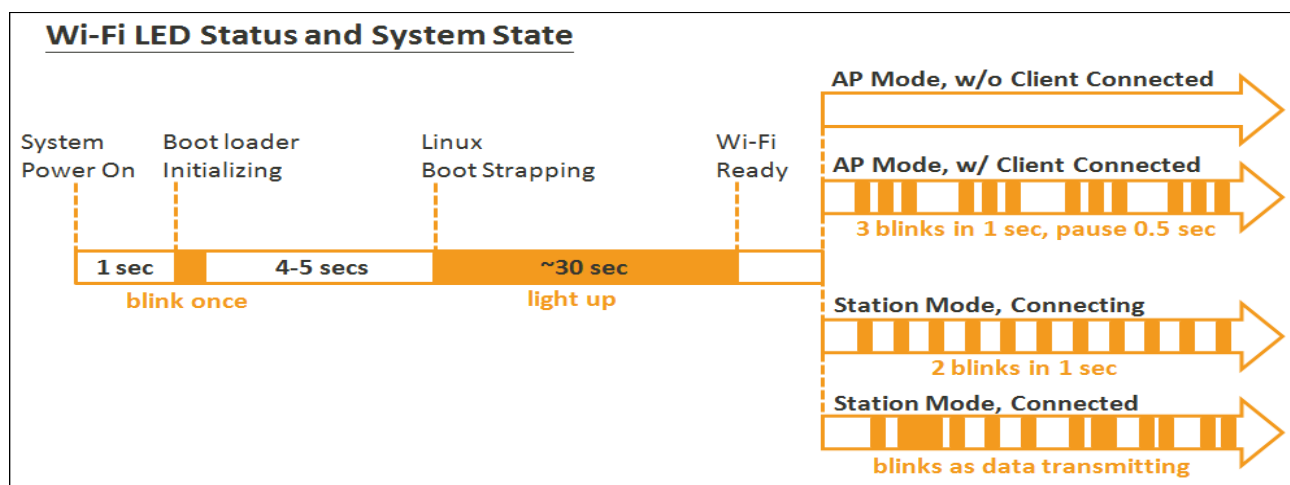
The Barracuda App Server, including tutorials and browser based development tools, comes pre-installed on a Linux powered LinkIt Smart 7688. The device, by default, starts as a Wi-Fi Access Point, and you may connect and access this device using any Wi-Fi enabled computer.

Step 1 — Power up your board using a micro-USB cable

Plug one end of a **Micro USB** cable into the **power connector of the LinkIt Smart 7688 Barracuda App Server board** and the other end of the cable to a USB power source, such as your computer, or a USB power adapter. Make sure you connect the cable to the USB Power connector, not the USB Host connector near the MPU reset button. The Power LED (Green) will light up first, followed by the Wi-Fi LED (Orange) which will blink once. Then, after about 4 to 5 seconds, the Wi-Fi LED will light up to indicate the bootloader has initialized.



Then the boot up process begins, which takes about 30 seconds. Next, the Wi-Fi LED turns off, which means the system is ready to accept a Wi-Fi connection. The figure below shows how the Wi-Fi LED status matches the system state.

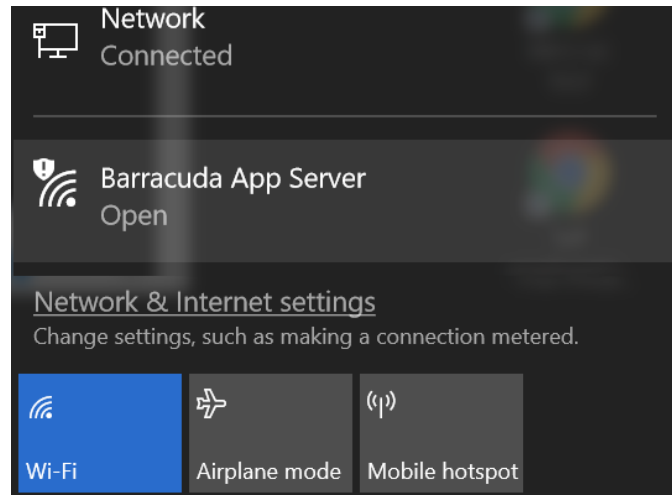


Step 2 — Connect your computer to the Barracuda App Server Access Point (AP)

Open the Wi-Fi connection utility on your computer and connect to the Access Point named **Barracuda App Server**.

You may use any type of computer including your phone and tablet, however, the computer should preferably have an external keyboard.

Note that some Android phones refuse to connect to networks that are not connected to the Internet. The Barracuda App Server Wi-Fi network, when operating as an Access Point, acts as a connection island and is not connected to the internet.



Step 3 — Access the Barracuda App Server's tutorials

In your browser, access <http://192.168.100.1>

You should see the page as shown to the right. Note that you may not be able to connect the browser immediately after connecting to the AP. Give the device a few seconds and try again if it fails.

You may alternatively navigate to <http://barracuda.local> if your computer has multicast Domain Name (mDNS) support. In Windows 7, you'll need to install the [Bonjour print service](#) because mDNS is not supported. This helps your computer discover the board's IP address within the local domain. For Windows 8 and later, Mac OS X, and Linux, mDNS is supported and you can use barracuda.local without additional software.

Follow the tutorials preferably from top to bottom as shown in the screenshot to the right.

Note: information specific to the LinkIt Smart has a yellow background.

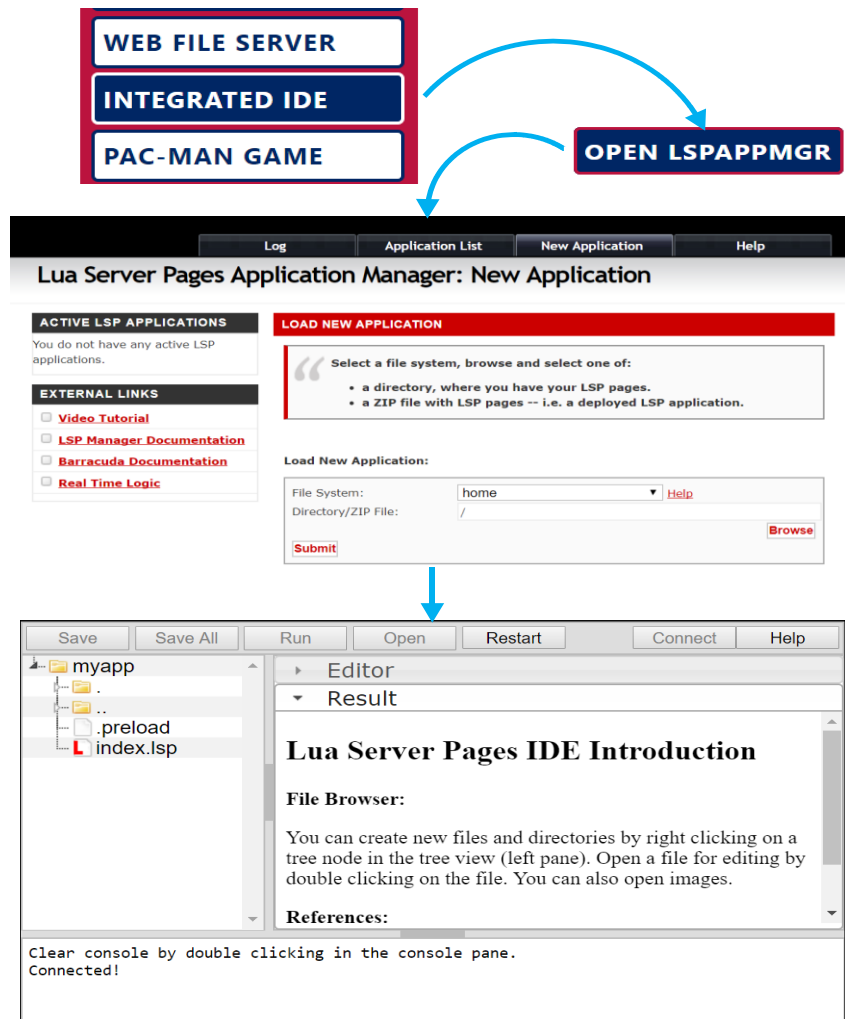


Development — Using the included web based LSP development environment

The Integrated Development Environment (IDE) provides a complete browser based development environment for LSP. With the IDE, you can create, edit, start, and stop LSP applications. Each active application can be managed using the LSP IDE.

Create and start working on a new LSP application as follows:

1. Click the INTEGRATED IDE button in the left pane, and click OPEN LSPAPPMGR.
2. Click the New Application Tab, select "Home" for file system, and click Browse.
3. Right click the top folder, create a new folder, and double click the folder to select it.
4. Click Submit to create a new application, and submit on the next page to use the default settings.
5. Click Start to start the new application.
6. Click the Edit button, which brings up the web based code editor.
7. Double click on the pre-generated index.lsp file and start working on your first LSP page.

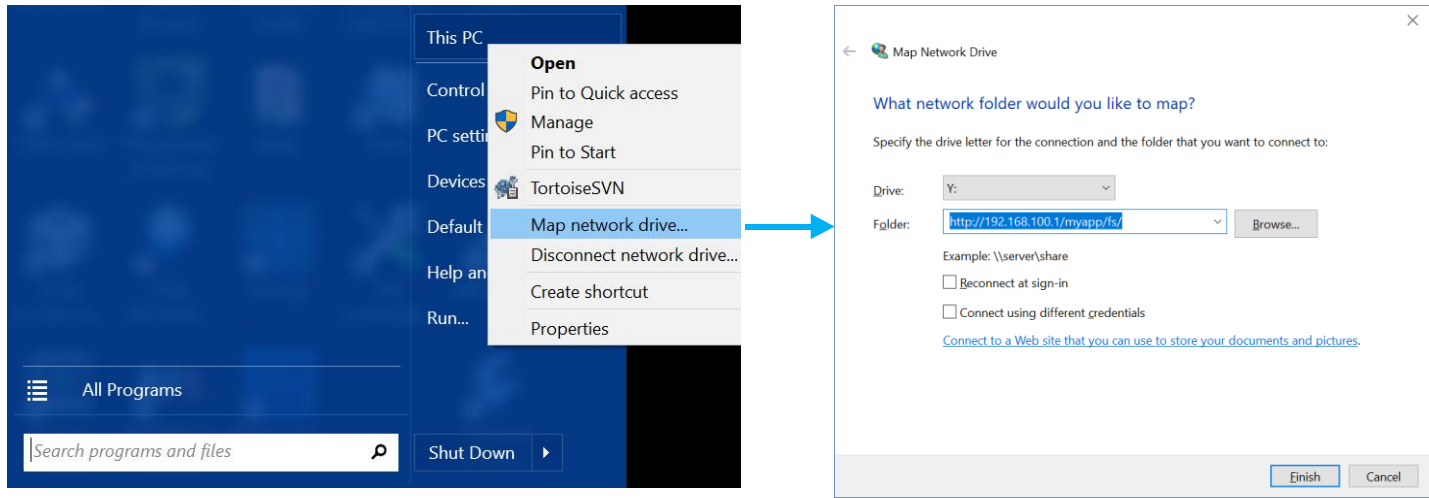


Development — Using an external editor

The LinkIt Smart's Linux file system can be mounted/mapped as a network drive. You may create and edit files directly on the device by using your preferred text editor as soon as you map a drive to the device.

The Barracuda App Server includes a WebDAV server, which enables mapping/mounting the LinkIt Smart device from your computer.

Most computers support WebDAV including Mac, Linux, and Windows. The following screenshot shows how to map a Windows drive to the pre-configured LSP application "myapp":



The Linkit Smart device is pre-configured with the two following WebDAV URLs:

http://192.168.100.1/fs/	Access to root of the Linux file system
http://192.168.100.1/myapp/fs/	Access to /home/mako/myapp/

The second URL is to a pre-configured LSP application available at the URL `http://192.168.100.1/myapp/`. You may mount this drive and use any text editor on your computer to create files and sub-directories on the Linkit Smart in the directory `/home/mako/myapp/`. As an example, creating the file `index.lsp` in this directory makes the file accessible via your browser using the URL: `http://192.168.100.1/myapp/`

Note that the Barracuda App Server runs as the Linux user "root". Be careful when mounting the device using the first URL above, which provides access to the root folder. You may brick the device should you delete or modify system files via the WebDAV mount. In addition, note that some WebDAV clients, including the one in Windows, starts traversing into sub directories on its own. This may temporarily freeze the server when WebDAV is attempting to access resources that are not designed to be opened remotely such as the Linux `/proc/` directory.

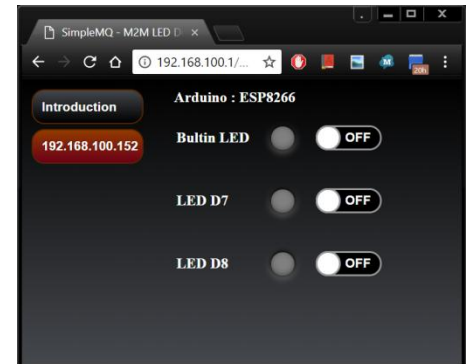
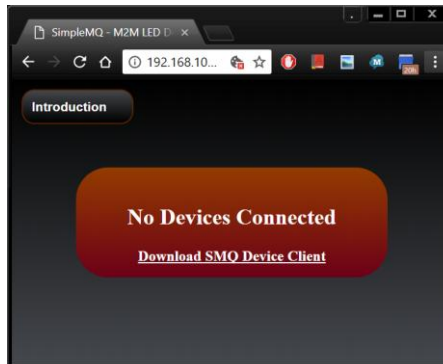
Development — Internet of Things (IoT) Tutorial

You may run the IoT demo/tutorial if your Barracuda App Server development kit comes with an ESP8266 Wi-Fi device. Start the demo as follows:

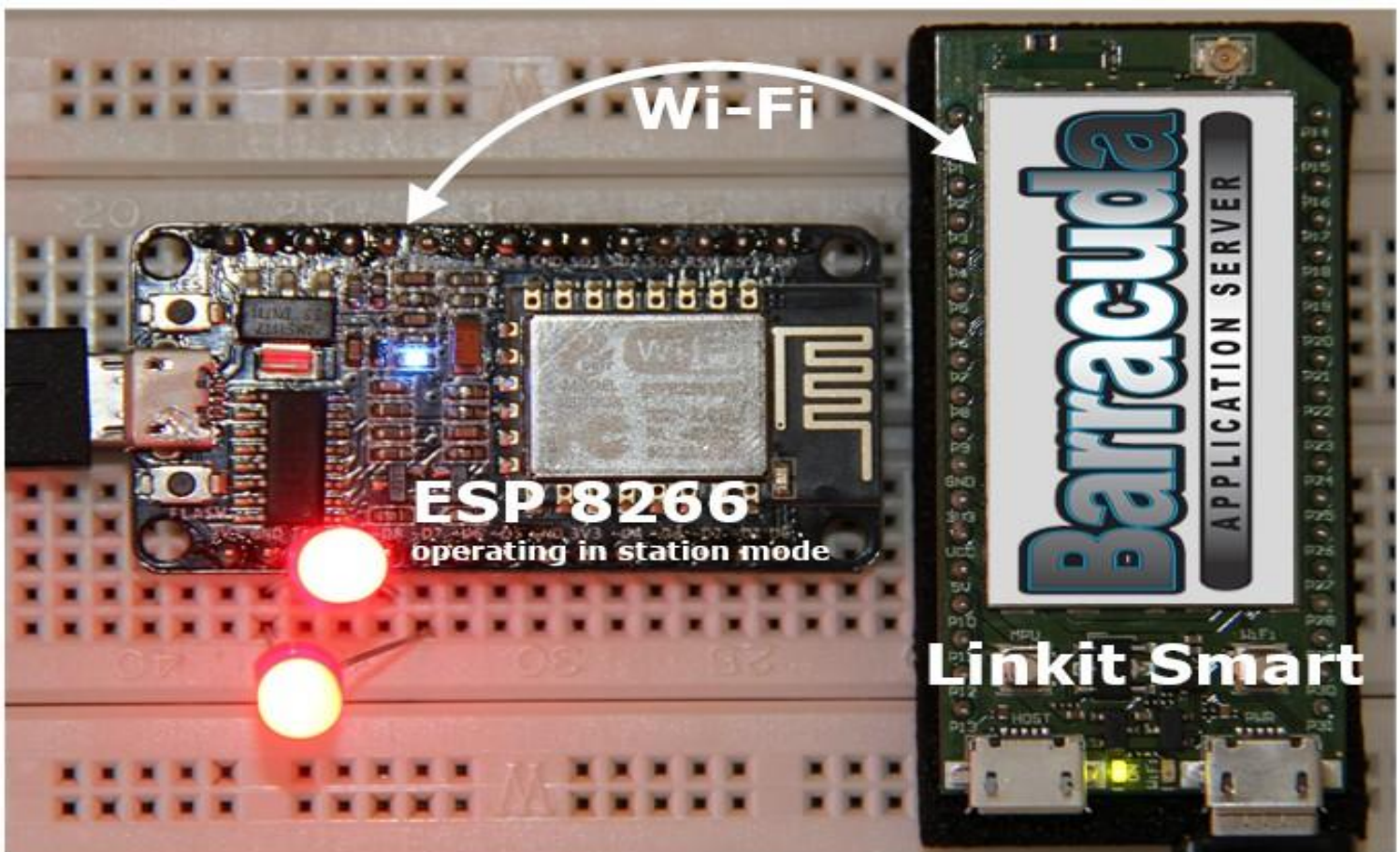
1. Click the INTERNET OF THINGS button in the tutorial's left pane followed by clicking the OPEN SMQ LED WEB INTERFACE button to bring up the SMQ LED demo's web page. Alternatively, navigate directly to the page by using the following URL: <http://192.168.100.1/smq/>
2. Plug one end of a Micro USB cable into the power connector of the ESP8266 and the other end of the cable to a USB power source.
3. The ESP8266 connects to the "Barracuda App Server" access point and registers with the SMQ broker that is part of the tutorials bundle. After 5 - 8 seconds, the device should show up in the SMQ LED demo's web page.

The two figures to the right show the SMQ LED demo's web page before and after powering on the ESP8266.

You may click the "Builtin LED" on/off button in the web interface to toggle the blue Built-In LED on the ESP8266 on and off.



You may optionally control the two other LEDs by connecting a LED between D7 and ground, and between D8 and ground. The following figure shows the ESP8266 hooked up to a breadboard with two LEDs connected.

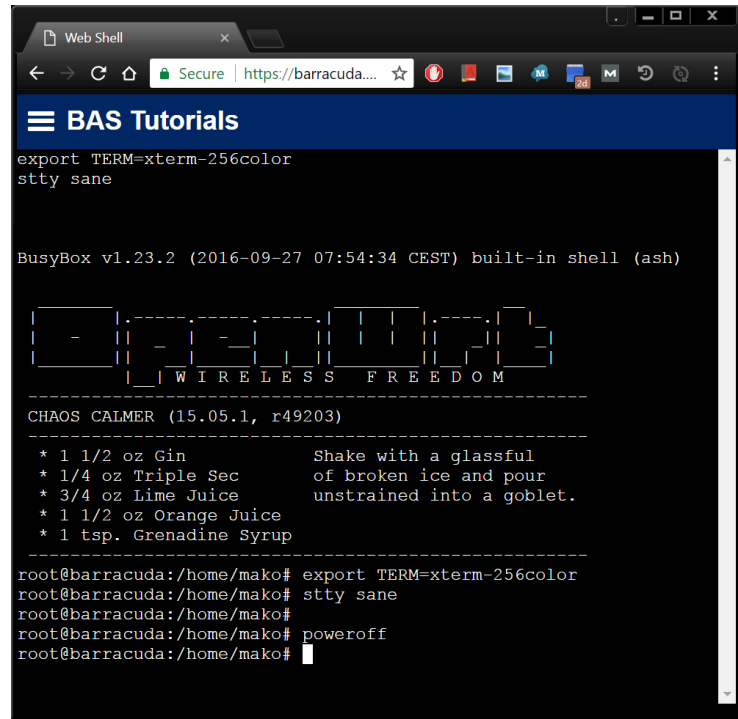


When done — Power off device

1. Click **WEB SHELL** in the left pane to bring up the browser based terminal console.
2. Enter the command **poweroff** and press the Enter key.
3. Wait a few seconds.
4. Remove the power supply.

WARNING:

The Web Shell provides a fully functional terminal in the browser. You may use any Linux command in the browser based terminal. Be careful if you are new to Linux, you may end up bricking the device.



Barracuda App Server Tutorials Components:

You may inspect the following data via the WEB SHELL by using standard Linux commands.

/bin/mako and /bin/mako.zip: The pre-compiled Barracuda App Server (mako) and its resource file (mako.zip) are stored in the /bin/ directory.

/home/mako - Tutorials and configuration files:

certs - SSL certificates used by the tutorials are stored in this directory.

Isppapps - All tutorials packaged as ZIP files are stored in this directory.

myapp - Directory for [development using an external editor](#).

mako.conf - Instructs Barracuda App Server (Mako Server) to load all tutorials and SSL certificates.

LED.so - pre-compiled shared library for the Lua to C Code tutorial.

SSL Certificates:

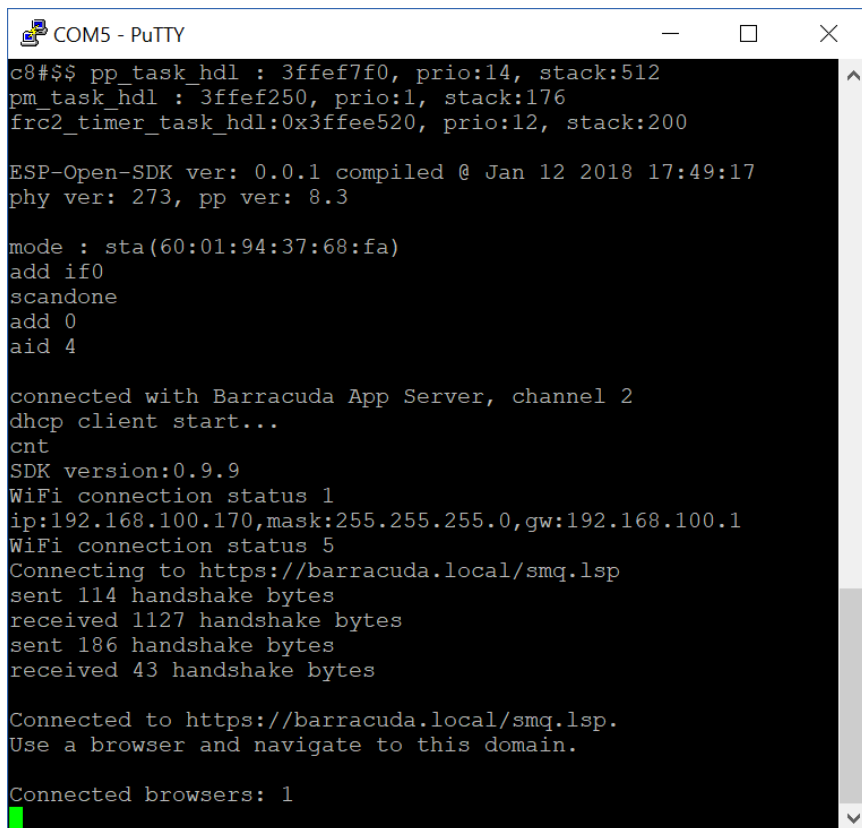
EC.barracuda.local.[key|pem] - Elliptic Curve certificate used when the ESP8266 connects to the Barracuda App Server.

RSA.barracuda.local.[key|pem] - RSA certificate used when navigating to the domain name "barracuda.local".

RSA.MakoServer.[key|pem] - - RSA certificate used when navigating to the domain name "makoserver".

ESP8266 information — a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability.

The ESP8266 is pre-flashed with our [SMQ LED demo](#), but instead of connecting to our [online SMQ cluster demo](#), the pre-flashed firmware is designed to connect to the access point "barracuda app server" and then to the domain name "barracuda.local". The SMQ client connects to the SMQ broker integrated in the Barracuda App Server tutorials.



```
COM5 - PuTTY
c8#$$ pp_task_hdl : 3ffef7f0, prio:14, stack:512
pm_task_hdl : 3ffef250, prio:1, stack:176
frc2_timer_task_hdl:0x3ffee520, prio:12, stack:200

ESP-Open-SDK ver: 0.0.1 compiled @ Jan 12 2018 17:49:17
phy ver: 273, pp ver: 8.3

mode : sta(60:01:94:37:68:fa)
add if0
scandone
add 0
aid 4

connected with Barracuda App Server, channel 2
dhcp client start...
cnt
SDK version:0.9.9
WiFi connection status 1
ip:192.168.100.170,mask:255.255.255.0,gw:192.168.100.1
WiFi connection status 5
Connecting to https://barracuda.local/smq.lsp
sent 114 handshake bytes
received 1127 handshake bytes
sent 186 handshake bytes
received 43 handshake bytes

Connected to https://barracuda.local/smq.lsp.
Use a browser and navigate to this domain.

Connected browsers: 1
```

The screenshot to the left shows the printouts from the ESP8266 when it boots up. You may connect any terminal program to the USB serial interface by powering the ESP8266 from one of your computer's USB connectors. We use [putty](#) and the ESP8266's USB-serial connected on COM5. The serial speed must be set to 115200.

Notice the TLS handshake messages in the screenshot to the left. We use SharkSSL and the secure version of SMQ called [SharkMQ](#) in the firmware. The complete Elliptic Curve Asymmetric TLS Handshake is surprisingly fast considering that the code executes through serial flash -- in other words, the code executes in place (XiP) through the SPI protocol. The ESP8266 board we are using has an external SPI flash chip.

The reason we use an Elliptic Curve certificate is to conserve RAM memory. The ESP8266 is configured according to our [Certificate Management for IoT](#) tutorial.

You may easily flash your own SMQ client by installing the [ESP8266 Arduino environment](#) and by downloading the non secure [SMQ client for Arduino](#).

FreeRTOS, lwIP, SharkSSL, and SharkMQ based firmware

Note that the pre-flashed firmware is not using the Arduino environment, but is instead based on the [FreeRTOS/lwIP SDK for ESP8266](#). Contact Real Time Logic if you are interested in building your own FreeRTOS, lwIP, and SharkSSL based firmware.