Doc Arduino Yun

Saturday, April 16, 2016 5:22 PM

INTRODUCTION

The purpose of this document is to describe how to configure two Arduino Yuns to operate autonomously, communicate over WiFi, connect to an Amazon Web Services IOT endpoint, and (by messaging one another through that intermediary) create a functional example of IOT on the public cloud.

This introduction gives the overall arc of the implementation, introducing the main terminology (actors) in the process. The terms are re-introduced below with more detail.

AWS IOT came out of Beta release circa 2015 and immediately provided the option of connecting an Arduino Yun. The Arduino family of microcontrollers have evolved as a well-supported environment for interfacing small computers to physical devices, both sensors and actuators. In the latter category we include things like displays. The Arduino Leonardo is distinguished by having a more powerful ATmega32u4 processor compared to the ubiquitous Arduino Uno; and the idea of the Arduino Yun is to add to that Leonardo processor a separate Linux processor with an abbreviated operating system ('Linino'), a Python 2.7 installation, WiFi and a lot of memory by means of an SD card. That second processor is by Atheros so in this document I refer to three components on the Yun as follows:

- Atmega (ATmega32u4) is the Arduino microprocessor
- Atheros (AR9331) is the Linux processor
- Bridge is an intermediary chip that provides serial communication between Atmega and Atheros.

This document proceeds in three stages: Basics, IOT and Operations. There are a lot of details to learn along the way and while we do go over them it is important to notice the LINKS section below. Here you will find various tutorials on doing essentially the same thing. Those resources may prove to be better guides than this one since this one was written 'in flight'.

Finally the operational task here is to have two physically adjacent Arduino Yuns equipped with photoresistors and laser diodes talk to one another as follows:

Yun 1: Turns on laser diode illuminating Yun 2 photoresistor.
Yun 2: Detects light, publishes a message to AWS 'Bright!'
IOT: Registers event from Yun 2
Yun 1: Checking IOT finds a directive 'Turn off laser'
Yun 1: Turns off laser
Yun 2: Publishes to AWS 'Dim…'
Yun 2: Turns on laser diode illuminating Yun 1 photoresistor
Yun 1: Detects light, publishes a message to AWS 'Bright!'

…and so on ad infinitum. What will the latency be of these two battling laser beams? It would be nice to see it come in at around one second; so place your bet now.

KEY IDEAS

- As we proceed there will be stumbling blocks. The first that I encountered is that the Arduino Yun (contrary to published documentation) first comes up on WiFi as 'Linino-XXXXXX' with a password of 'doghunter'. Once you update the Atheros OS and connect to local WiFi this changes to 'Arduino-XXXXXX' with password 'arduino' as advertised. I try to capture as many of these details as possible here.

- You must install a considerable amount of hardware and software per the various tutorials; some of which may prove to be redundant or alternate-path. This includes (where I work from a Windows PC):
  - Arduino Yun (available for about USD60 in 2016; here we work with two units)
  - USB cables
    - Normal to Micro
    - Your cables must support full USB connectivity
    - WARNING many USB cables are inadequate and fail to connect Yun to PC
  - PuTTY (to log in to the Atheros)
  - WinSCP (to copy files to the Atheros)
  - Arduino IDE (to program the Atmega: Should be a circa 2016 or later version)
  - AWS IOT Arduino SDK (library to allow the Atmega sketch to direct the conversation with AWS IOT)
  - On Atheros:
    - opkg update
    - opkg install distribute
    - opkg install python-openssl
    - opkg install openssh-sftp-server (enables sftp from PC to Atheros)
  - AWS CLI (Command Line Interface to the AWS console; runs from Windows Console or PowerShell)
- You will find in PROCEDURE 1 that it is not necessarily trivial to connect to your Yun over WiFi. For this you need its ip address which you can obtain from the Atmega by installing and running the WiFiStatus example sketch. This is after you have configured the Yun to connect to your ambient WiFi. This is why having a good USB cable is essential; without it you cannot program the Atmega and you are stuck. If you are on a home WiFi and can log on to your router you can also find ip addresses for Yuns on the router interface page.

LINKS

Arduino

- http://wiki.linino.org/doku.php?id=wiki:gettingstarted (linino wiki: somewhat helpful; see below)
- http://www.arduino.cc/en/Guide/ArduinoYun "Getting Started with the Arduino Yun"
- http://www.arduino.org/products/boards/4-arduino-boards/arduino-yun
- http://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/arduino-Yun-schematic.pdf (schematic)
- http://labs.arduino.org/Arduino%20Yun Arduino Yun Lab with many examples of getting the Yun to do stuff

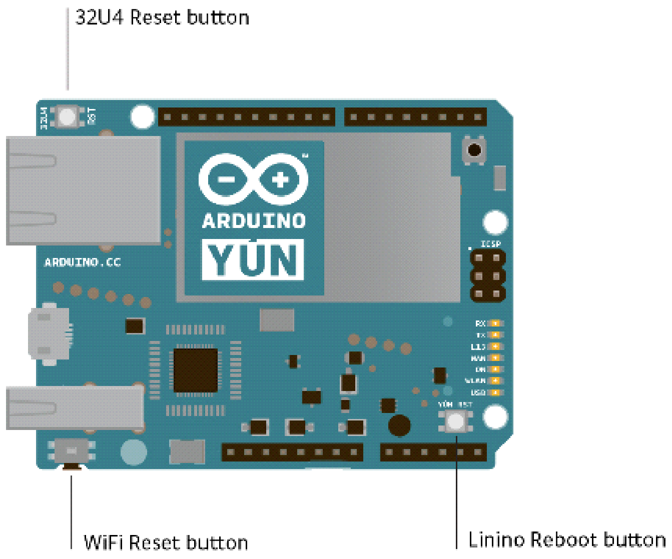
AWS

- https://aws.amazon.com/blogs/aws/category/aws-iot/ Overview of the AWS IOT platform
- https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html
- https://docs.aws.amazon.com/iot/latest/developerguide/iot-quickstart.html AWS IOT Quick Start
- https://us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/connect?thing=Ishmael On AWS once you create a Yun thing and generate a certificate and keys you are directed to this useful page which explains how to start connecting.
- https://github.com/aws/aws-iot-device-sdk-arduino-yun/blob/master/README.md GitHub readme on how to connect an Arduino Yun to AWS. You are sent here after creating an IOT endpoint at AWS (plus the cert and keys).
- http://ajays-tech.com/blog-1/2016/1/6/arduino-yun-and-aws-iot A tutorial to try first (before this one)

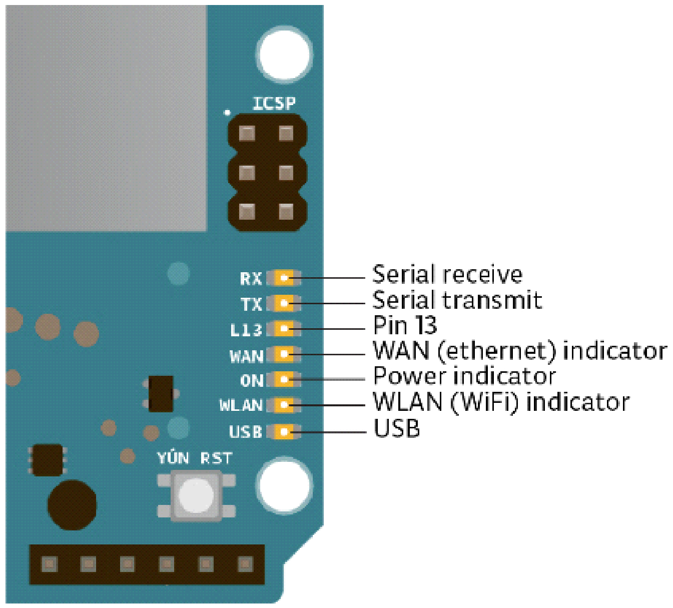
PROCEDURAL 1: BASICS

To begin using the Yun I installed the Arduino IDV and (through a microUSB connector) powered one up, found its WiFi network, connected to that, and using a browser went to the URL 192.168.240.1 to connect to its Linino interface. Before launching ahead on the same plan I recommend reading these qualifying remarks on this process.

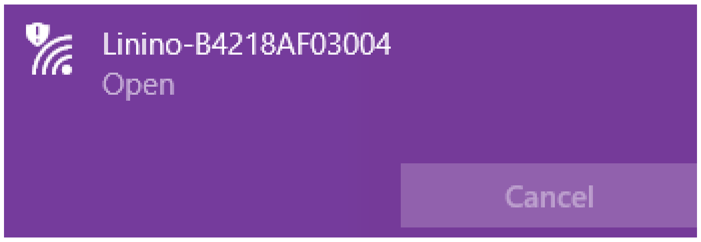
The Yun has three reset buttons; including the WLAN button which is sideways-mounted near the large USB port.



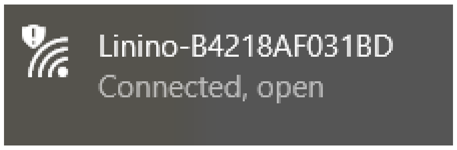
The Yun also has a set of status lights and when it boots up -- a process that takes over a minute -- the WiFi light eventually comes up white and you should be able to see it over WiFi at that point.



The Yun's own WiFi access port should appear on a nearby computer; in my case


Linino-B4218AF03004
Open

Leading to...

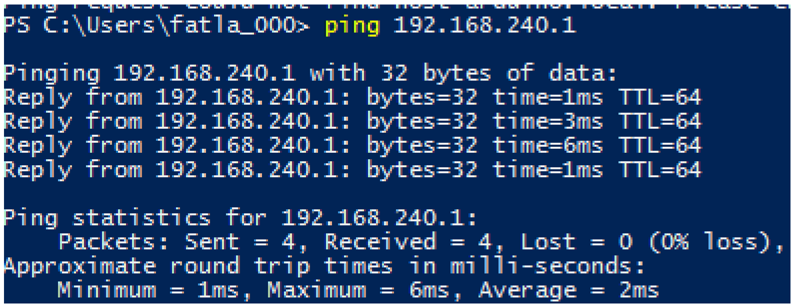

Linino-B4218AF031BD
Connected, open

The documentation online seems confused on a couple of points: When it first boots you see 'Linino' and it is only after logging in (password is 'doghunter') and doing some configuration stuff in the browser that the proper 'Arduino' network appears.

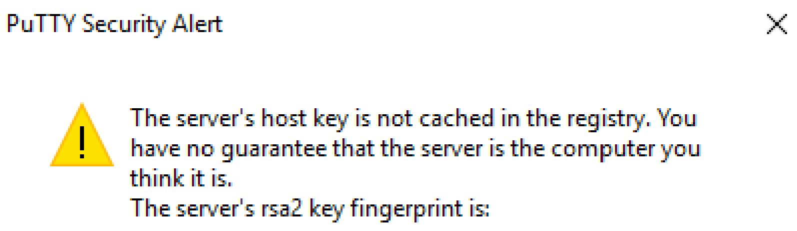
The correct help for navigating through this Linino startup is http://wiki.linino.org/doku.php?id=wiki:gettingstarted

So the the details: Once you log on (enable cookies in your browser) you can configure the Arduino Yun. To get a handle on this the working idea is that the Yun has two processors onboard.

From PowerShell it is possible to ping the Yun:

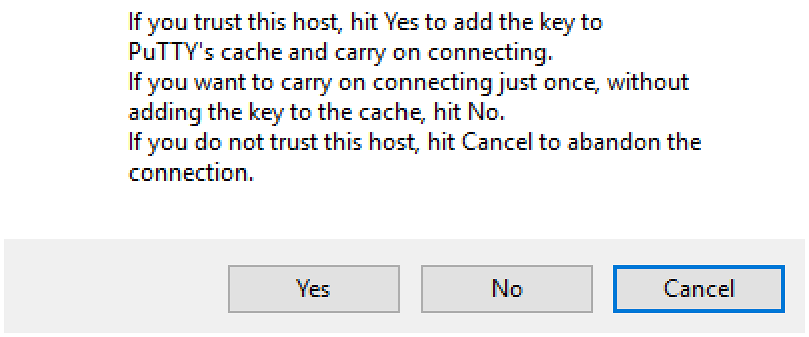

```
Ping request could not find host arduino.local. Please c
PS C:\Users\fatla_000> ping 192.168.240.1

Pinging 192.168.240.1 with 32 bytes of data:
Reply from 192.168.240.1: bytes=32 time=1ms TTL=64
Reply from 192.168.240.1: bytes=32 time=3ms TTL=64
Reply from 192.168.240.1: bytes=32 time=6ms TTL=64
Reply from 192.168.240.1: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.240.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 6ms, Average = 2ms
```

Using putty I can ssh to 192.168.240.1 with this speedbump:


PuTTY Security Alert

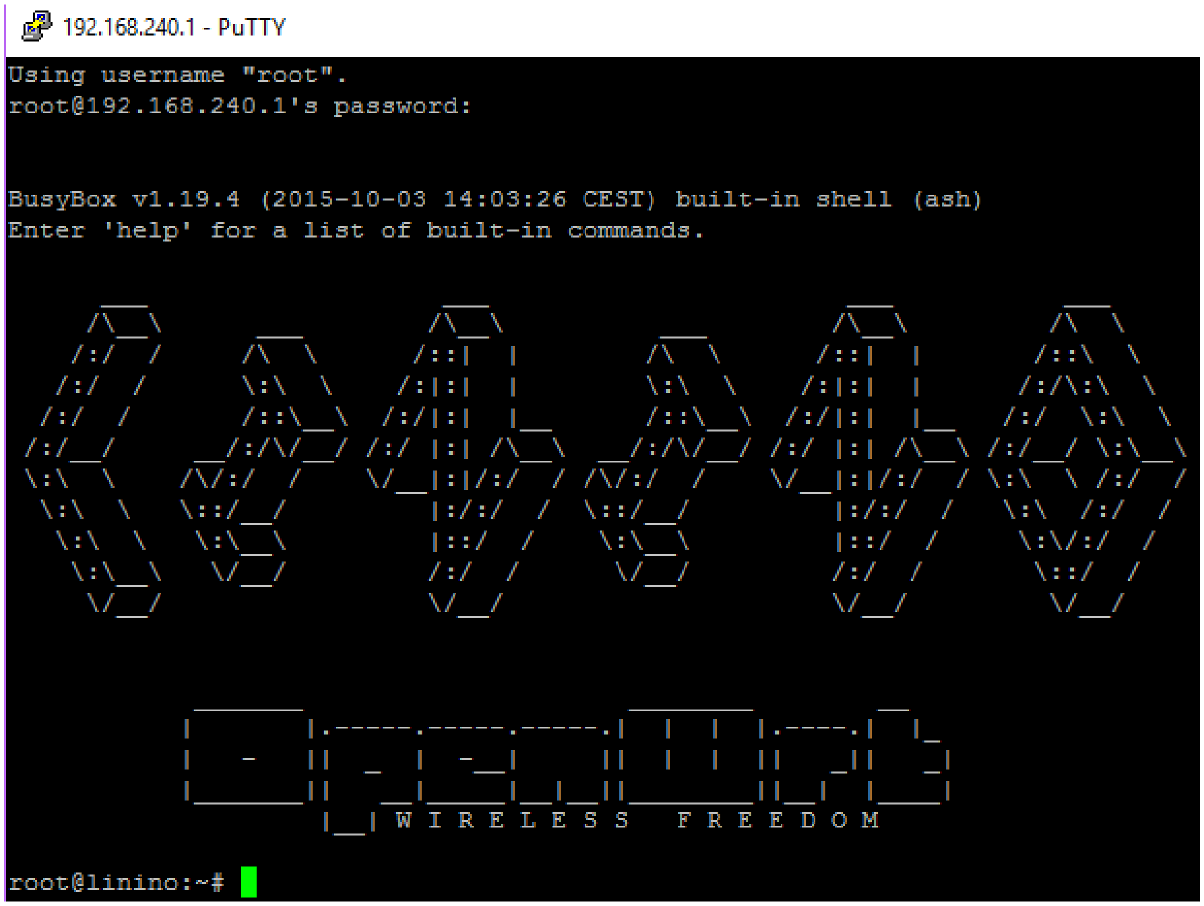
The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:

[redacted]



If you trust this host, hit Yes to add the key to
PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without
adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the
connection.

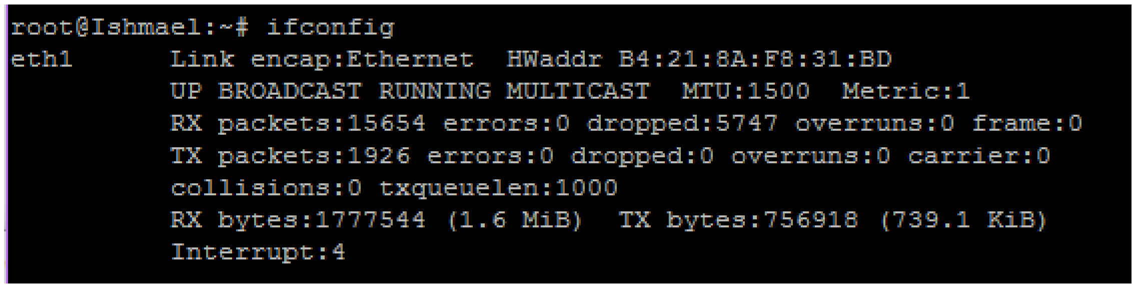
Remember the password is 'doghunter' and you get to here:



```
Using username "root".
root@192.168.240.1's password:


BusyBox v1.19.4 (2015-10-03 14:03:26 CEST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

     ___                   ___                       ___           ___
    /\__\      ___        /\__\          ___        /\__\         /\  \
   /:/  /     /\  \      /::|  |        /\  \      /::|  |       /::\  \
  /:/  /      \:\  \    /:|:|  |        \:\  \    /:|:|  |      /:/\:\  \
 /:/  /       /::\__\  /:/|:|  |__      /::\__\  /:/|:|  |__   /:/  \:\  \
/:/__/     __/:/\/__/ /:/ |:| /\__\  __/:/\/__/ /:/ |:| /\__\ /:/__/ \:\__\
\:\  \    /\/:/  /    \/__|:|/:/  / /\/:/  /    \/__|:|/:/  / \:\  \ /:/  /
 \:\  \   \::/__/         |:/:/  /  \::/__/         |:/:/  /   \:\  /:/  /
  \:\  \   \:\__\         |::/  /    \:\__\         |::/  /     \:\/:/  /
   \:\__\   \/__/         /:/  /      \/__/         /:/  /       \::/  /
    \/__/                 \/__/                     \/__/         \/__/


            _______                     ________        __
           |       |.-----.-----.-----.|  |  |  |.----.|  |_
           |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
           |_______||   __|_____|__|__||________||__|  |____|
                    |__| W I R E L E S S   F R E E D O M

root@linino:~#
```

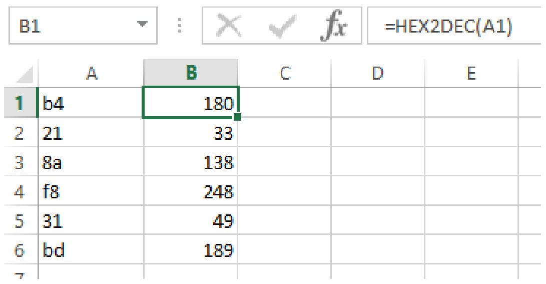
So that is command line access. I am hopping onto the Yun via WiFi, not a USB cable.

From the command line I issue 'ifconfig' to try and get an IP address that I can use for a direct connect over Ethernet... Here is the result for Ishmael, one of my two Yuns, for eth1 (which the website says is the correct ethernet port):



```
root@Ishmael:~# ifconfig
eth1      Link encap:Ethernet  HWaddr B4:21:8A:F8:31:BD
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15654 errors:0 dropped:5747 overruns:0 frame:0
          TX packets:1926 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1777544 (1.6 MiB)  TX bytes:756918 (739.1 KiB)
          Interrupt:4
```

Notice there is no 'inet addr'... but what I am looking for is an IP address that looks like 192.168.240.1. For the record I can import the HWaddr value above as hex into Excel and use HEX2DEC to convert to decimal, like this:


B1 =HEX2DEC(A1)

	A	B	C	D	E
1	b4	180			
2	21	33			
3	8a	138			
4	f8	248			
5	31	49			
6	bd	189			

This leads nowhere however; and what is more there is no /etc/network/interfaces file with this information; so things are rather mysterious. Only 'ifconfig' has worked although 'lshw' (which reads like ls-hardware) is supposed to be enlightening as well for Ubuntu. Then there are additional commands ip and route.

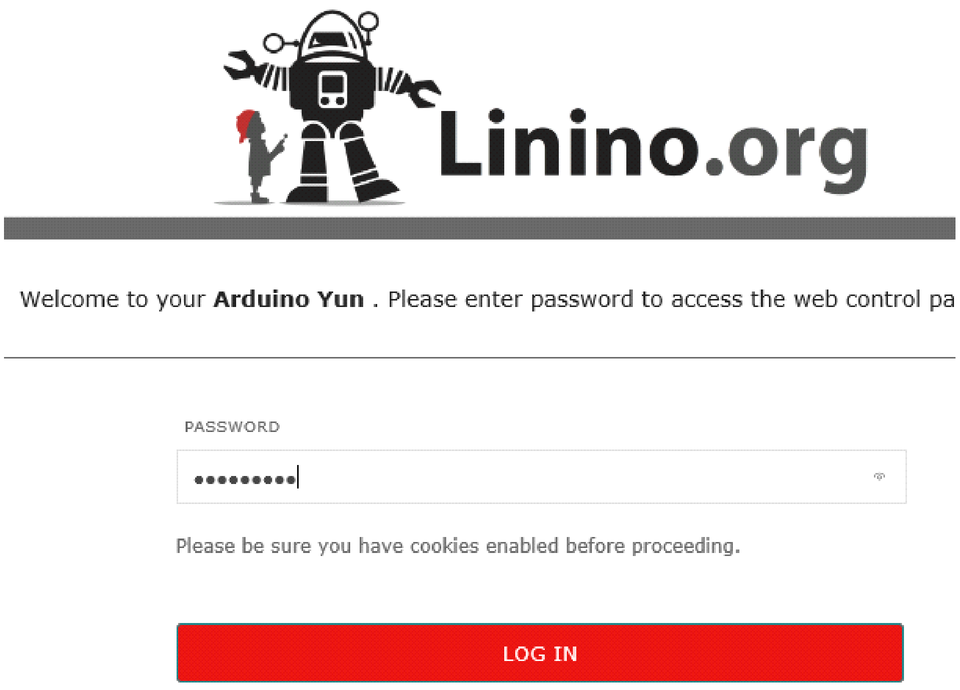
```
# route add default gw 10.0.0.1 eth1
# route -n
```

Then there is the /etc/resolve.conf file that can hold temporary DNS entries. And then there is also dhcp; another access protocol (?)... so this is a difficult situation and reminds us of the Slipper Slope Maxim:

Slippery Slope Maxim: When things start going wrong and I get stuck trying to follow a recipe (embedded devices in particular) I hereby vow to get off the slippery slope and follow these steps:

1. Step back, get some coffee, take a break.
2. Back up to the last point at which things were working correctly
3. Look for a simple obvious way forward and ***do that***.

There is also the configuration process through the browser. The first thing to do (per the Wiki) is to load the OS upgrade file onto a micro SD card and plug that into the Yun before you boot. Then in the browser, on the Linino network, once you enter that hard-to-find 'doghunter' password on the Linino login (192.168.240.1):

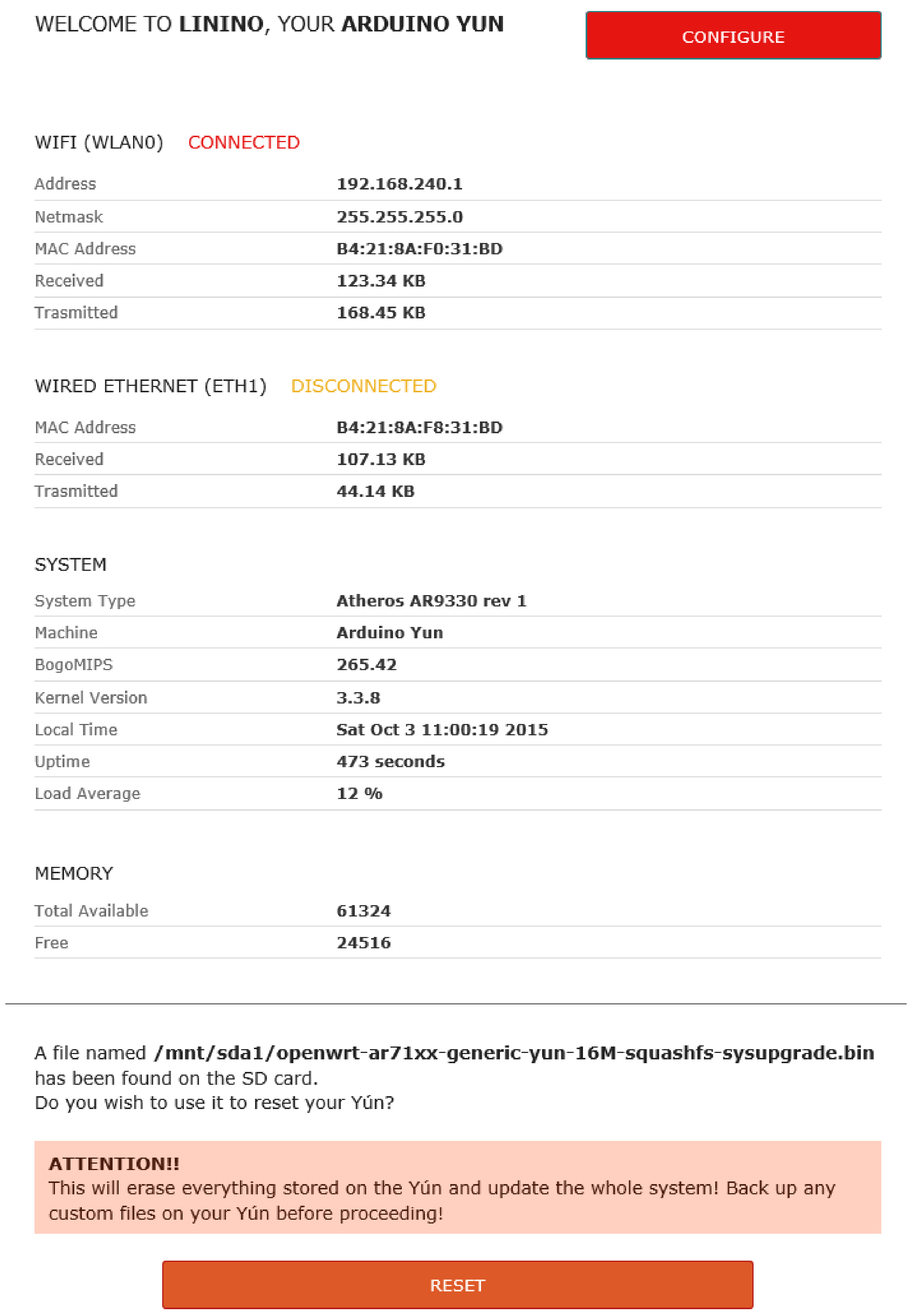

Welcome to your **Arduino Yun** . Please enter password to access the web control pa

PASSWORD

•••••••••

Please be sure you have cookies enabled before proceeding.

And then you get this auto-detect of the upgrade file you have placed on the SD Card:


WELCOME TO **LININO**, YOUR **ARDUINO YUN**

WIFI (WLAN0) CONNECTED

Address	**192.168.240.1**
Netmask	**255.255.255.0**
MAC Address	**B4:21:8A:F0:31:BD**
Received	**123.34 KB**
Trasmitted	**168.45 KB**

WIRED ETHERNET (ETH1) DISCONNECTED

MAC Address	**B4:21:8A:F8:31:BD**
Received	**107.13 KB**
Trasmitted	**44.14 KB**

SYSTEM

System Type	**Atheros AR9330 rev 1**
Machine	**Arduino Yun**
BogoMIPS	**265.42**
Kernel Version	**3.3.8**
Local Time	**Sat Oct 3 11:00:19 2015**
Uptime	**473 seconds**
Load Average	**12 %**

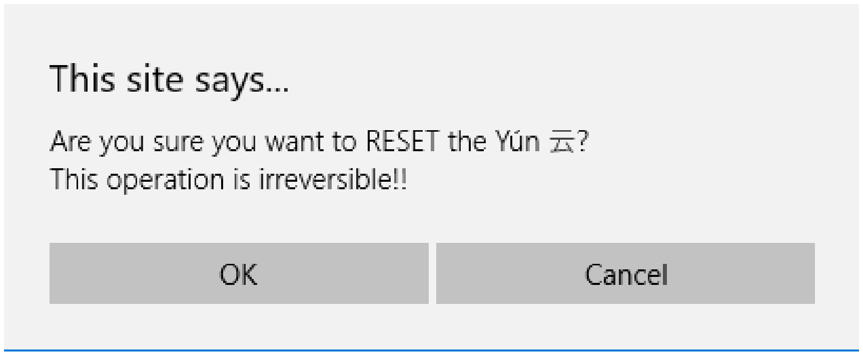
MEMORY

Total Available	**61324**
Free	**24516**

A file named **/mnt/sda1/openwrt-ar71xx-generic-yun-16M-squashfs-sysupgrade.bin** has been found on the SD card.
Do you wish to use it to reset your Yún?

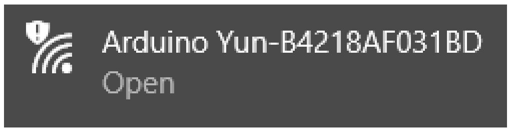
ATTENTION!!
This will erase everything stored on the Yún and update the whole system! Back up any custom files on your Yún before proceeding!

So in case you skipped over this up above: Before going to configure the Yun I am going to run this 'RESET' to upgrade the operating system. I pulled in the latest upgrade from Arduino onto a microSD card; and this is installed in the Yun SD card reader so that's why I get to run this now. It warns me after I click RESET:

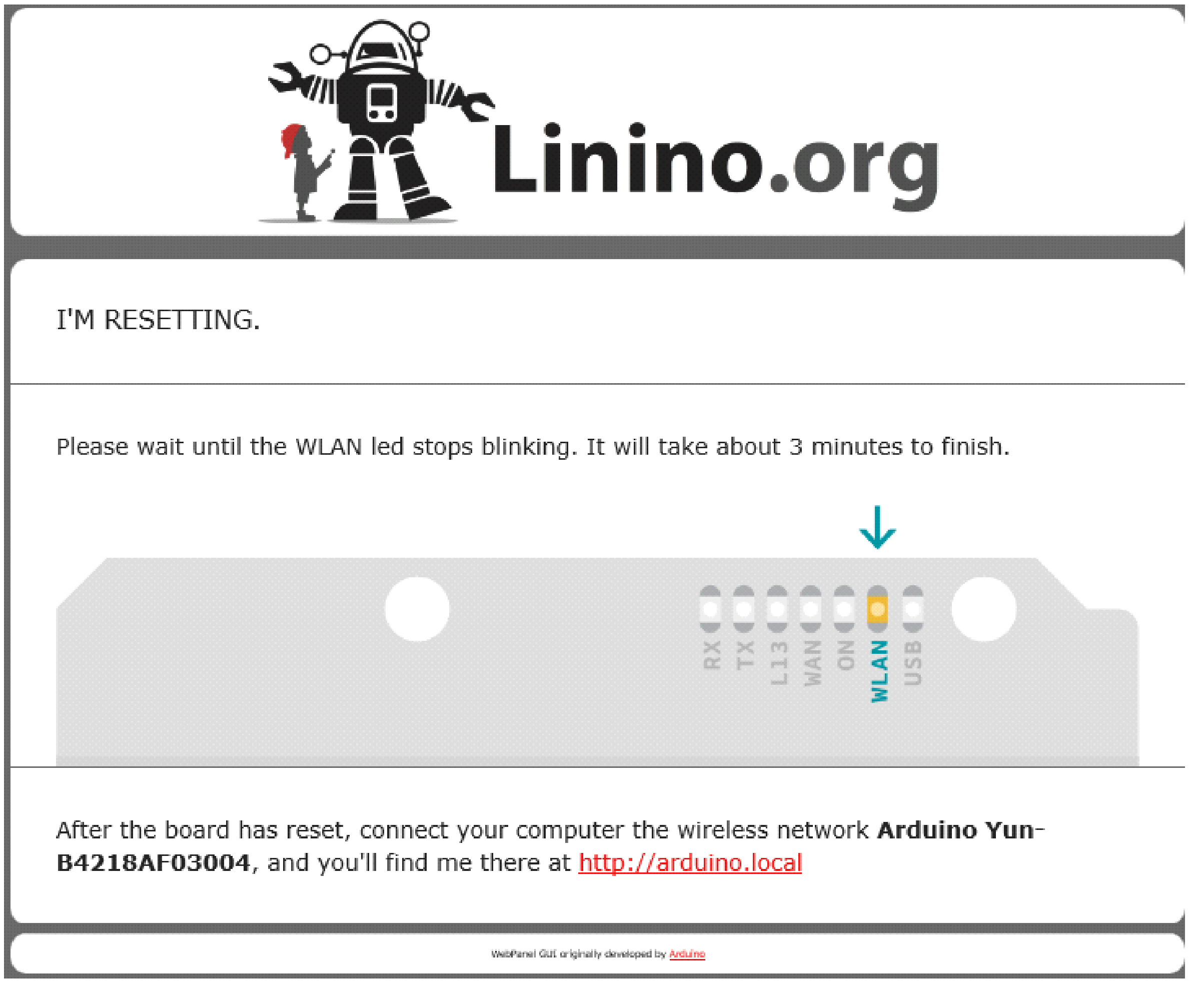

This site says...

Are you sure you want to RESET the Yún 云?
This operation is irreversible!!

Now I reboot and the WLAN is correctly Arduino:


Arduino Yun-B4218AF031BD
Open

And now the password is no longer 'doghunter'. Now it is 'arduino' as you will find in most of the documentation about the Arduino Yun; so we had to do the Linino/doghunter rigamarole before we got back to where the documentation is correct. After the RESET I see this:



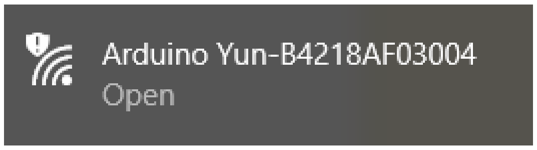
I'M RESETTING.

Please wait until the WLAN led stops blinking. It will take about 3 minutes to finish.

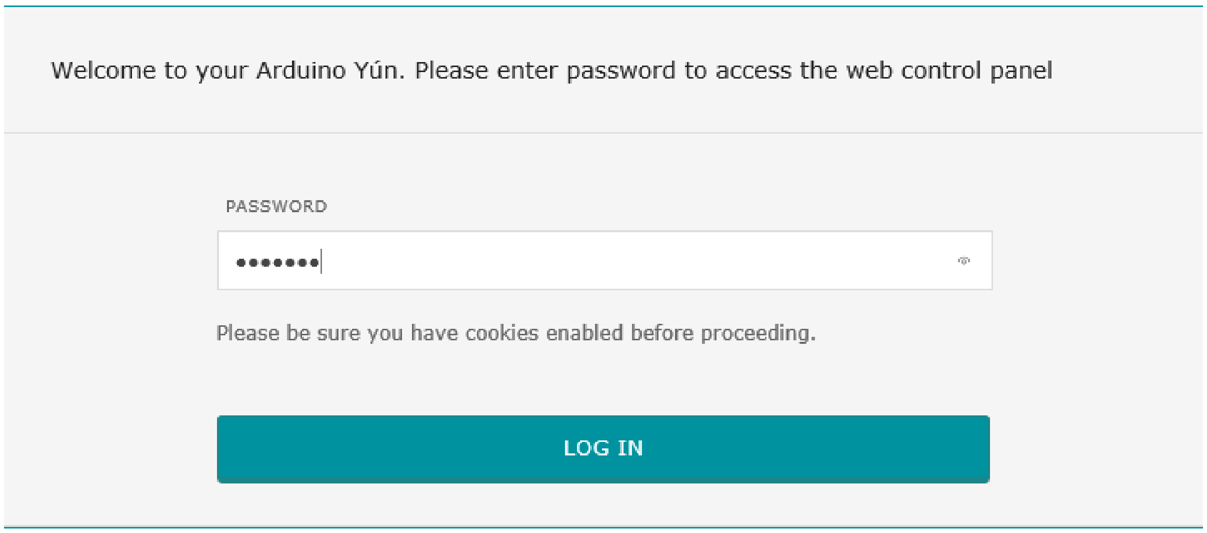
After the board has reset, connect your computer the wireless network **Arduino Yun-B4218AF03004**, and you'll find me there at http://arduino.local

WebPanel GUI originally developed by Arduino

And sure enough now we have:


Arduino Yun-B4218AF03004
Open

So I connect to that and now from the same URL 192.168.240.1 notice the login looks different, password is 'arduino':

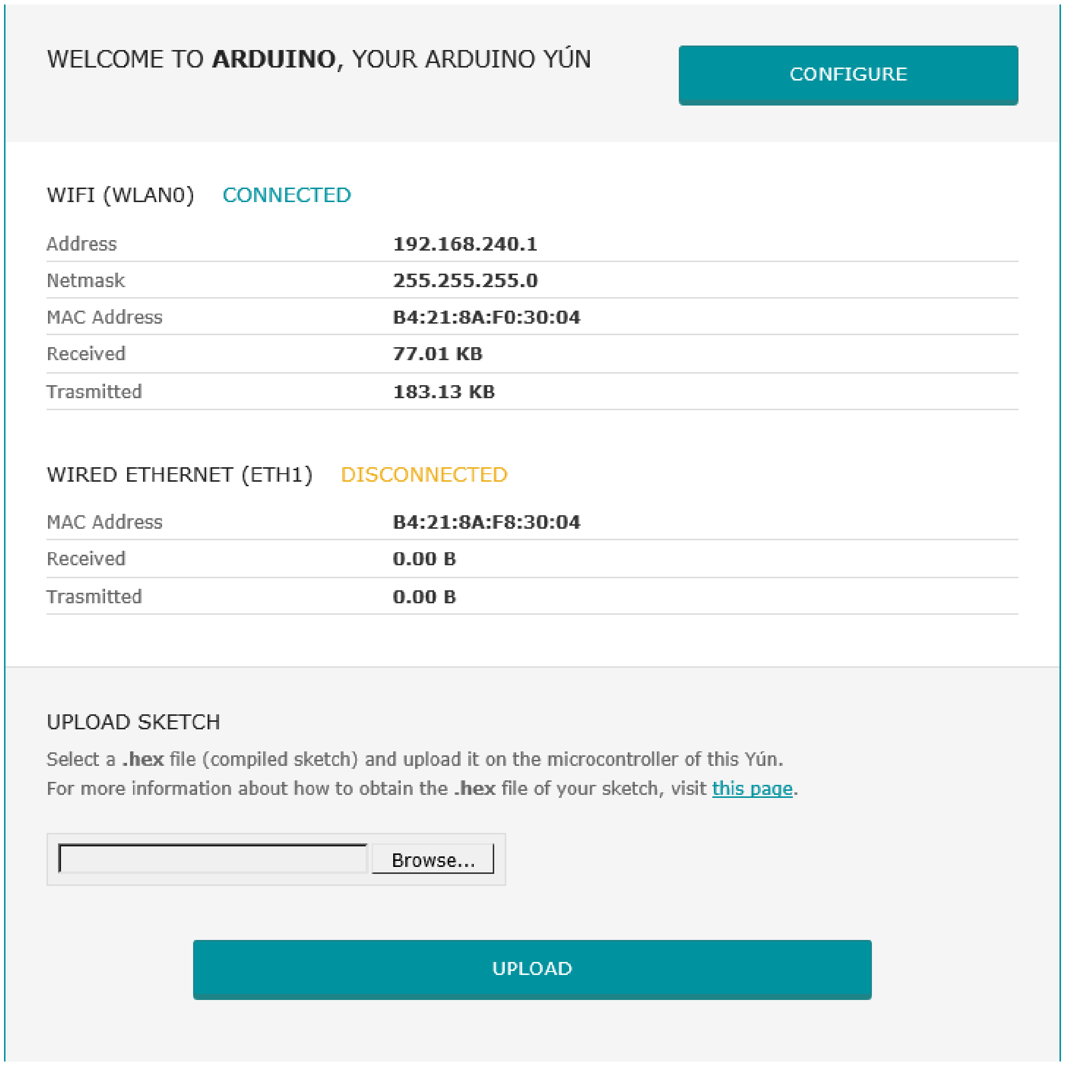

Welcome to your Arduino Yún. Please enter password to access the web control panel

PASSWORD

•••••••

Please be sure you have cookies enabled before proceeding.

So then let's take a look at the entry page:


WELCOME TO **ARDUINO**, YOUR ARDUINO YÚN

WIFI (WLAN0) CONNECTED

Address	192.168.240.1
Netmask	255.255.255.0
MAC Address	B4:21:8A:F0:30:04
Received	77.01 KB
Trasmitted	183.13 KB

WIRED ETHERNET (ETH1) DISCONNECTED

MAC Address	B4:21:8A:F8:30:04
Received	0.00 B
Trasmitted	0.00 B

UPLOAD SKETCH

Select a **.hex** file (compiled sketch) and upload it on the microcontroller of this Yún.
For more information about how to obtain the **.hex** file of your sketch, visit this page.

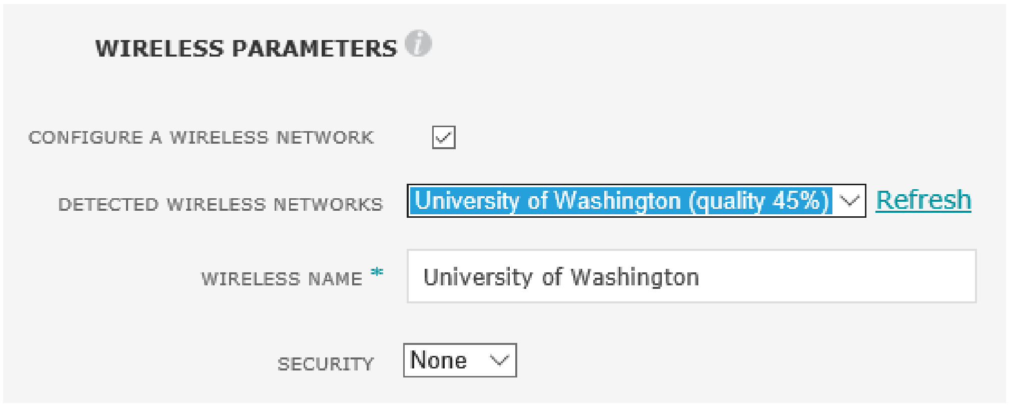
So notice we can upload sketches (Arduino code) here. But first let's go for the CONFIGURE process:

I renamed my Yuns:

Queequeg
ishmael

Time Zone = America Los Angeles.

I configured the Arduino to connect to the UW open WiFi:


WIRELESS PARAMETERS

CONFIGURE A WIRELESS NETWORK ✓

DETECTED WIRELESS NETWORKS: University of Washington (quality 45%)

WIRELESS NAME *: University of Washington

SECURITY: None

Then RESET. At this point take a coffee break; we have gotten through the initial slog. Here are a few basics to reflect on as we get ready to move forward:

- The Yun has three reset buttons: One for the WiFi and one for each of the two processors.
- There are some important things we want to be able to do now:

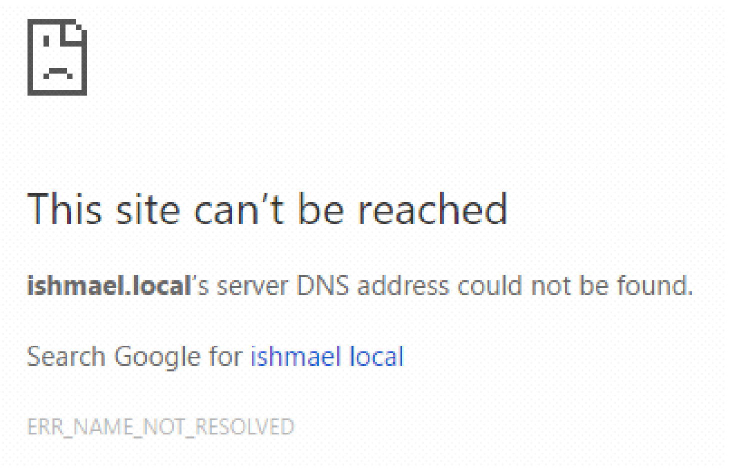
1. ssh into our Yuns to work on them like Linux systems (if that is our need for the project… likely!)
2. Copy files to the Yun using WinSCP (secure copy) from the laptop PC
3. Change the Yun WiFi network to connect to a local network for which we have access
    a. Hold down the WiFi reset button with the Yun powered up for > 5 seconds (not more than 30)
    b. Connect to the Yun at 192.168.240.1 using your browser; log in; click [Configure]
    c. Use the auto-scan in the configuration panel to select the local WiFi and RESET the Yun
    d. Reconnect your laptop to the local WiFi while the Yun boots…
    e. See the section below on reconnecting to the Yun.
    f.
4. Avoid getting the "Do you want to upgrade your operating system?" prompt
    a. Wipe the OS upgrade from the SD card you have installed in the Yun
5. Connect to the Yun via a USB serial cable
    a. If you plug in the Yun and your laptop does not confirm (e.g. with a jingle that says a new device was plugged in) you may have run into the same problem I did: Bad USB cables. Try some others if you do not see the COM port showing up in the IDE. The other possibility is that you need to update the FTDI drivers for the Arduino. You can find them by searching online.
6. Hard Ethernet: Not covered here.

Reconnecting to the Yun

Once you move the Yun to the local WiFi connection: Its new ip address can be difficult to identify and even after you identify it it might not connect. You can generally find the ip address using the Example WiFiStatus sketch but I have had mixed success getting to that ip address (i.e. connecting wirelessly to the Yun) at UW.

The WiFiStatus example sketch on the Yun (under 'Bridge') will show the current ip address once you open the serial window (after burning the program into the Atmega). This should get you ssh and WinSCP access to the Yun.

Here is what happens when you try using <name>.local if things are not working properly:



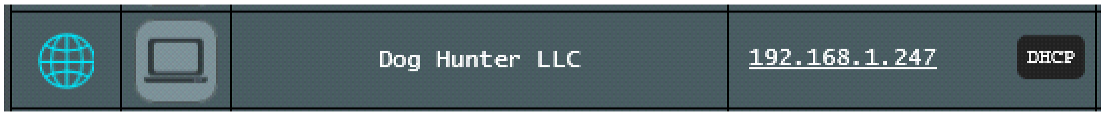
This site can't be reached

ishmael.local's server DNS address could not be found.

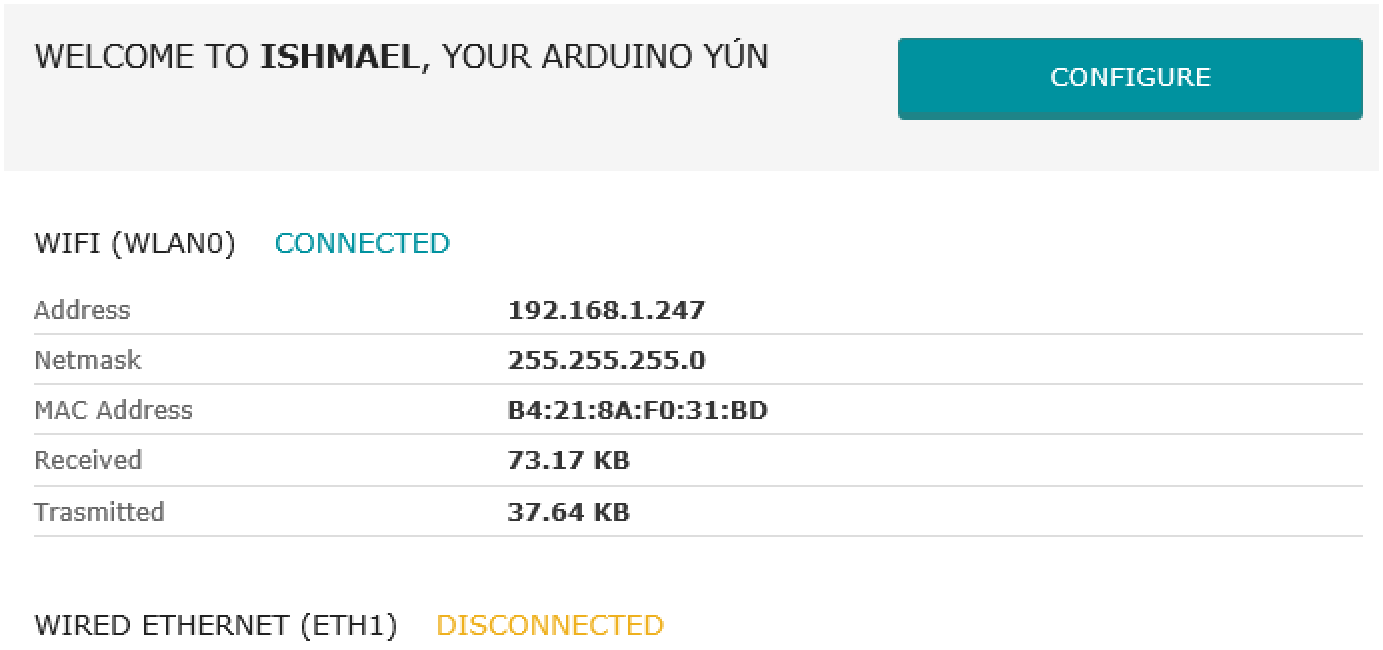
Search Google for ishmael local

ERR_NAME_NOT_RESOLVED

Another approach: On your home WiFi router you can log in and see the Yun listed with an ip address, as in here: 192.168.1.247…



Once you point your browser to your Yun (Ishamael in my case) you can see its ip address validated:



WELCOME TO **ISHMAEL**, YOUR ARDUINO YÚN

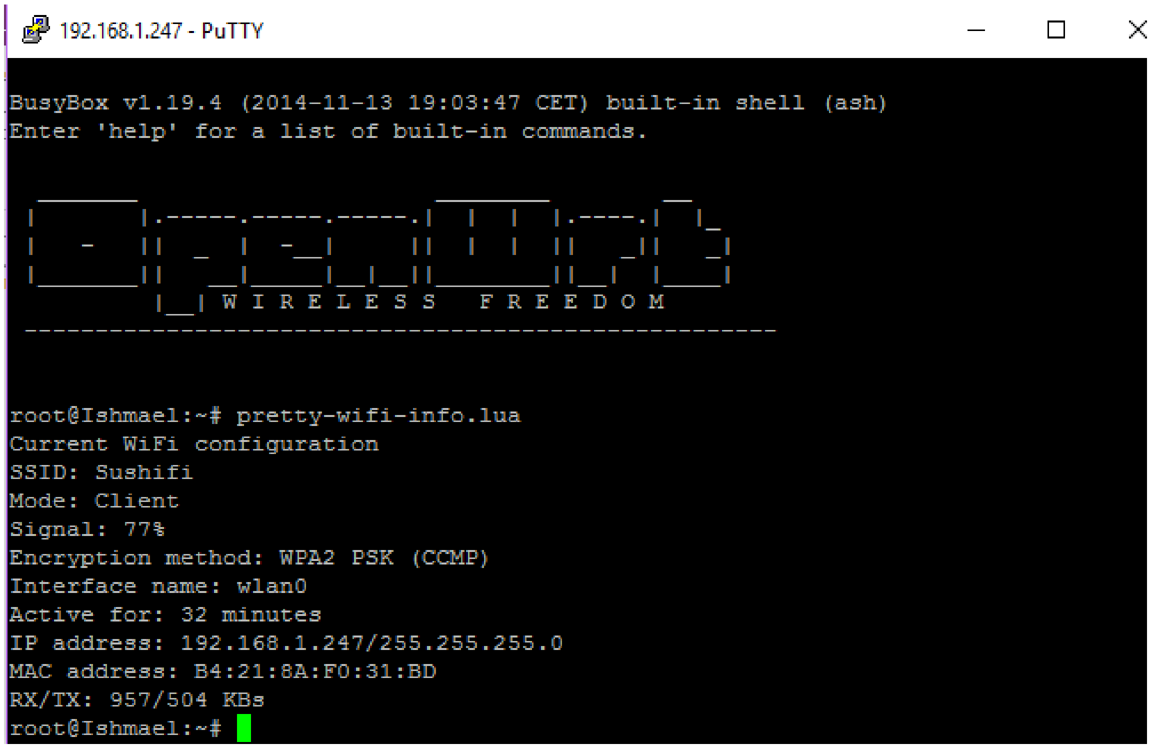
WIFI (WLAN0) CONNECTED

Address	192.168.1.247
Netmask	255.255.255.0
MAC Address	B4:21:8A:F0:31:BD
Received	73.17 KB
Trasmitted	37.64 KB

WIRED ETHERNET (ETH1) DISCONNECTED

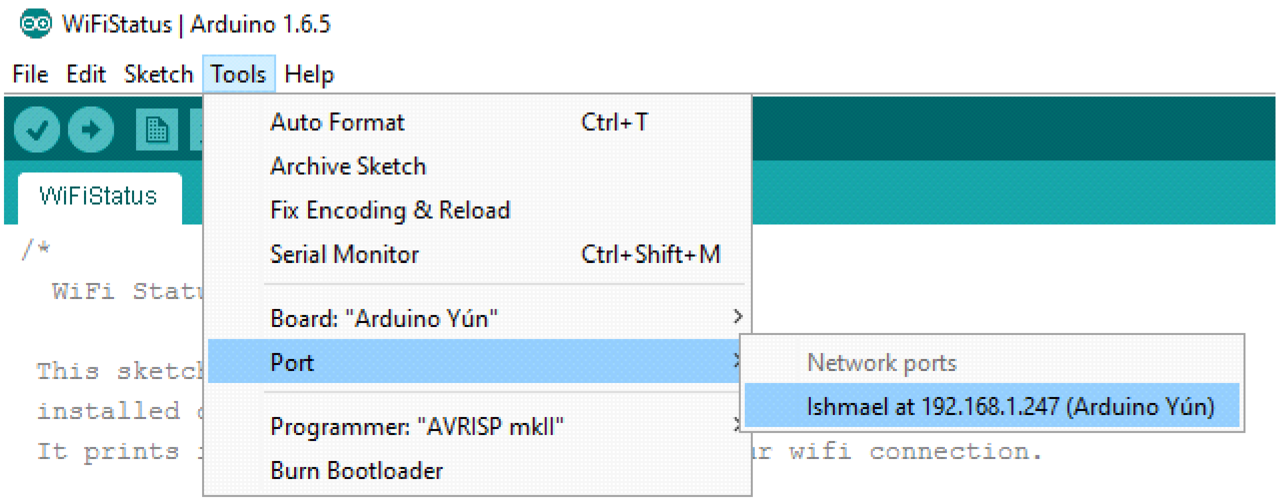
When you ssh to the Yun you may encounter a Security Alert as shown above. This is normal and you can just click on Yes to move through.

On the PuTTY terminal window I issue the same shell command that WiFiStatus uses (sending it from the Atmel through the Bridge to the Atheros):



```
192.168.1.247 - PuTTY

BusyBox v1.19.4 (2014-11-13 19:03:47 CET) built-in shell (ash)
Enter 'help' for a list of built-in commands.

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------

root@Ishmael:~# pretty-wifi-info.lua
Current WiFi configuration
SSID: Sushifi
Mode: Client
Signal: 77%
Encryption method: WPA2 PSK (CCMP)
Interface name: wlan0
Active for: 32 minutes
IP address: 192.168.1.247/255.255.255.0
MAC address: B4:21:8A:F0:31:BD
RX/TX: 957/504 KBs
root@Ishmael:~#
```

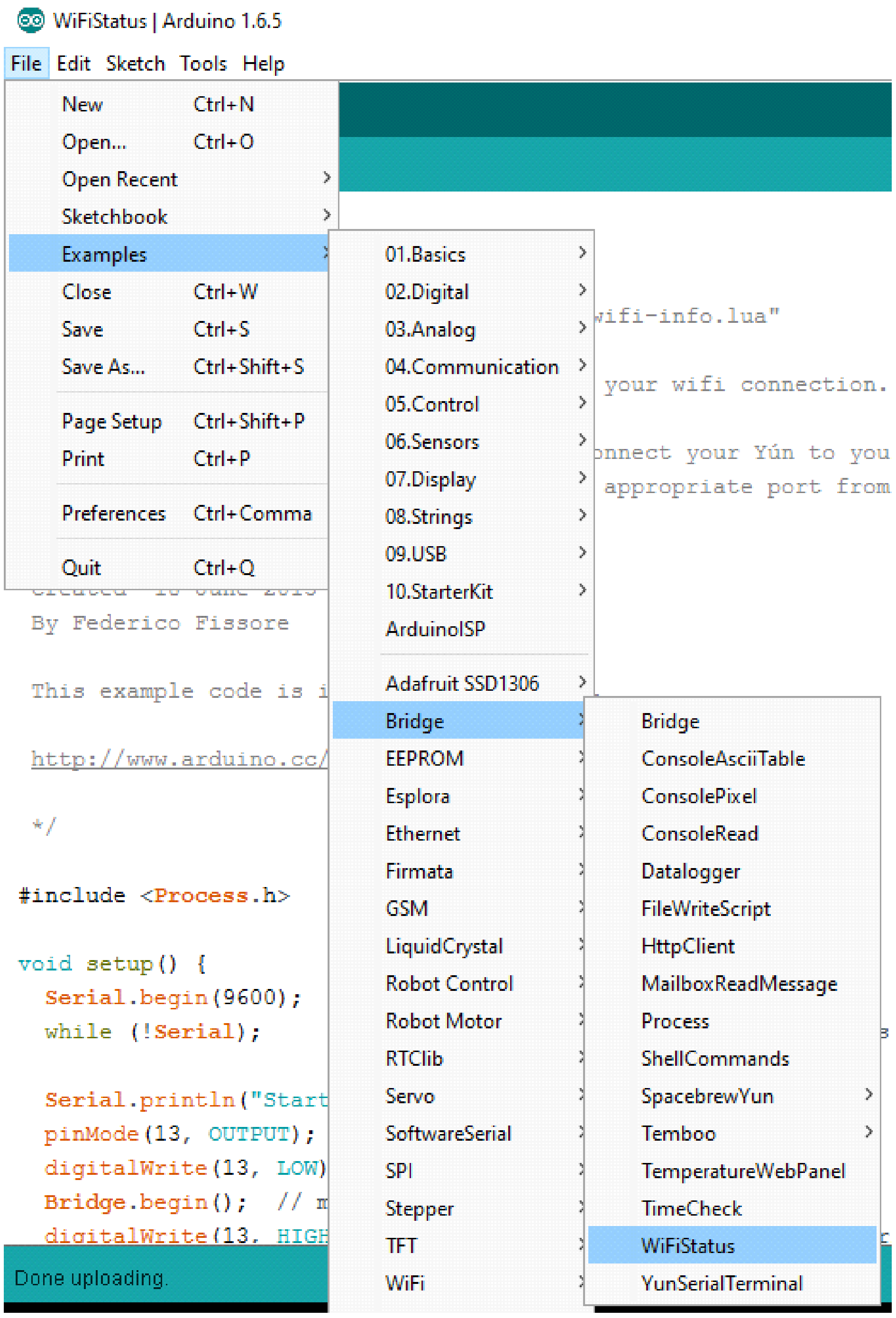
The Yun now registers properly as well in the Arduino IDE:



```
/*
  WiFi Statu
 This sketch
 installed 
 It prints                      ur wifi connection.
```

However in the above screen capture you can see that 'Network ports' is greyed out. This means I cannot load a sketch directly into the Yun via the USB cable. This was because my USB cable was deficient. I looked around and found a cable that worked and the COM port appeared in the Arduino IDE.

Here is the location of the WiFiStatus sketch in the 'Bridge' Example sketches. Again this is strictly going into the Atmega, not the Atheros.



```
wifi-info.lua"
 your wifi connection.
onnect your Yún to you
 appropriate port from

  By Federico Fissore

  This example code is i

  http://www.arduino.cc/

 */

#include <Process.h>

void setup() {
  Serial.begin(9600);
  while (!Serial);

  Serial.println("Start
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW)
  Bridge.begin();  // m
  digitalWrite(13, HIGH
```

Three brief remarks on uploading this or any sketch to the Yun microcontroller:

1. Sometimes on my Windows machine the COM3 serial port connection doesn't work properly (even though it is listed; see above) and I get some orange error text. I find that this is often fixed by going into the Tools menu > Port and selecting the Yun's entry.
2. The other 'fail to load' orange text for a well-tested sketch like this is possibly having the Serial Window open which I believe interferes with the transmission of the new program to the Atmega.
3. Third: At one point while uploading a sketch the IDE asked me to give the Yun root password. This is an authentication measure for crossing the Bridge to access the Atheros from the Atmega; so it is analogous to using ssh. To review: These three devices Atmega, Bridge and Atheros should be thought of as connected in series; the Bridge is a piece of hardware that enables the other two to communicate.

All that this Sketch does is run the command I showed above on the Linux board (henceforth the

Atheros):

# pretty-wifi-info.lua

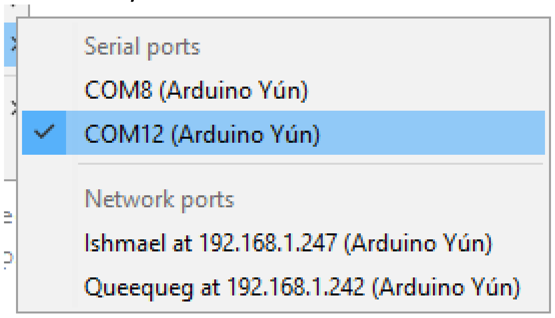
The Atmega gets the results back so they can be viewed in the serial window.

This is the WiFiStatus sketch output from the serial terminal:

```
Current WiFi configuration
SSID: University of Washington
Mode: Client
Signal: 72%
Encryption method: None
Interface name: wlan0
Active for: 2 minutes
IP address: 69.91.145.18/255.255.254.0
MAC address: B4:21:8A:F0:31:BD
RX/TX: 7/8 KBs
```

On my home network: I have both Ishmael and Queequeg Yuns powered off of separate USB ports.

The IDE says:

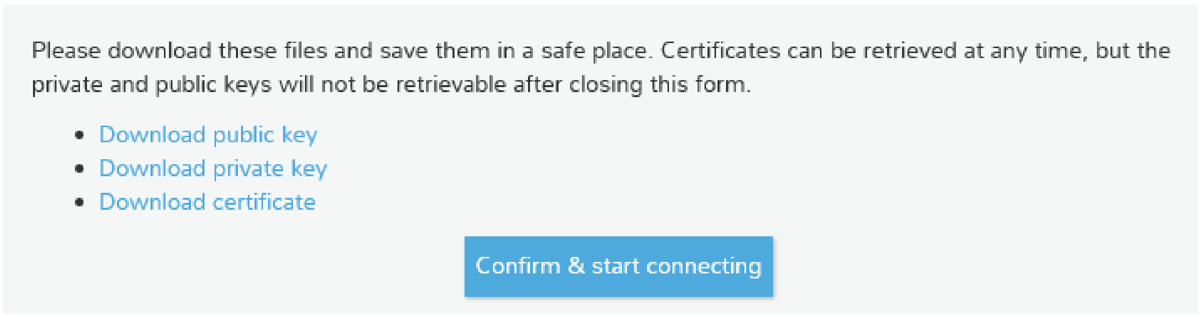

Serial ports
COM8 (Arduino Yún)
✓ COM12 (Arduino Yún)
Network ports
Ishmael at 192.168.1.247 (Arduino Yún)
Queequeg at 192.168.1.242 (Arduino Yún)

So with that let's declare victory and move on to AWS connectivity before we come back to the Yuns and get them to do something fun.

PROCEDURAL 2: Yun Cloud IOT

192.168.1.247 is Ishmael for this run.
192.168.1.242 is Queequeg
Home network

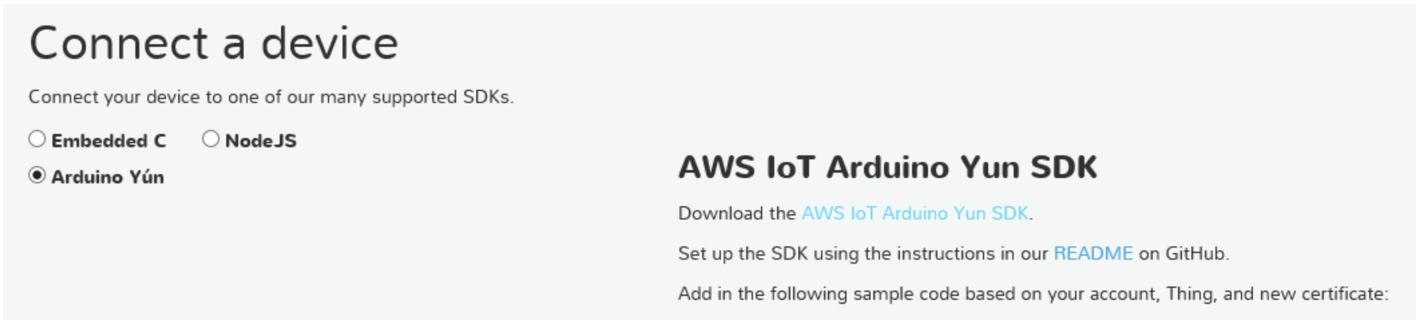
Connecting to AWS
I created an AWS IOT Endpoint called 'quack' which I connect to an Arduino Yun:



Please download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys will not be retrievable after closing this form.

- Download public key
- Download private key
- Download certificate

I created another and (after downloading those three files) I got this set of instructions from the 'Confirm' button shown above:



Connect a device

Connect your device to one of our many supported SDKs.

Embedded C [ ] NodeJS [ ] Arduino Yún [✓]

AWS IoT Arduino Yun SDK

Download the AWS IoT Arduino Yun SDK.

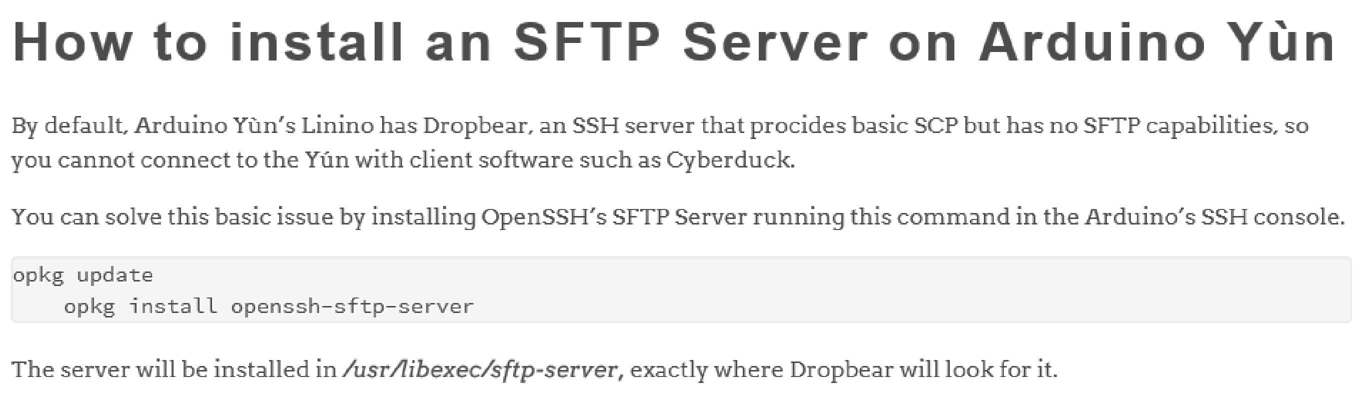
Set up the SDK using the instructions in our README on GitHub.

Add in the following sample code based on your account, Thing, and new certificate:

Installing the SDK also requires downloading a cert file ('.pem'?) from
https://www.symantec.com/content/en/us/enterprise/verisign/roots/VeriSign-Class%203-Public-Primary-Certification-Authority-G5.pem

I named it something else... we'll see if I need to name it '203-Public-Primary-Certification-Authority-G5.pem'.

I installed PuTTY already, I now installed WinSCP. But I can not connect to the Yun with WinSCP. It took about an hour to figure out that the Yun does not support sftp; it has to be installed. Ok...



How to install an SFTP Server on Arduino Yùn

By default, Arduino Yùn's Linino has Dropbear, an SSH server that procides basic SCP but has no SFTP capabilities, so you cannot connect to the Yún with client software such as Cyberduck.

You can solve this basic issue by installing OpenSSH's SFTP Server running this command in the Arduino's SSH console.

```
opkg update
    opkg install openssh-sftp-server
```

The server will be installed in */usr/libexec/sftp-server*, exactly where Dropbear will look for it.

So with that done let's reprise the installs that must be done on each Yun:

opkg update
opkg install distribute
opkg install python-openssl
opkg install openssh-sftp-server

Incidentally I am doing everything Twice because I have two Yuns (Ishmael and Queequeg). This is reminding me of what a pain it is to figure out how to do things across WiFi on a set of computers because as you figure things out you have to remember to go back and do the same thing on the other computer. Therefore: If at all possible just do One and document it; then do the next and see if you can go according to the documentation; and so on. Much better than bouncing back and forth.

This is particularly dangerous because Ishmael and Queequeg have different certs and keys. These must be hand-copied into the Python folder of the SDK before using WinSCP to move it over. The first time I did it I copied Queequeg's stuff onto Ishmael… A more effective method might be to create IOT thing folders each with their copy of the SDK; so then the cert/key files can be written individually into the Python/certs folder of each.

I am skipping the Web Socket step ('7') this time around; I may have to come back to it…

I did go back and rename the cert file because of this:

```
#define AWS_IOT_ROOT_CA_FILENAME "root-CA.crt"
// your root-CA filename
```

It is renamed on both Yuns.

PROCEDURAL 3: OPERATIONS

Now that we have two Yuns talking to the Amazon cloud it is time to do just a little bit more work to make them do something amusing. How about dueling lasers?

NOTES

- AVR Arduino Microcontroller
    - ATmega32u4
    - Input Voltage: 5V
    - 20 Digital I/O Pins
    - 7 PWM Channels
    - 12 ADCs
    - 16MHz Clock Speed
    - 32 KB Flash Memory
- Linux Microprocessor
    - Atheros AR9331
    - Operating Voltage: 3.3V
    - Architecture: MIPS @400MHz
    - Ethernet: IEEE 802.3 10/100Mbit/s
    - WiFi: IEEE 802.11b/g/n
    - PoE Compatible 802.3af
    - USB Type-A 2.0 Host
    - Micro-SD Card Reader
    - 64 MB DDR2 RAM
    - 16 MB Flash Memory