ESP8266 is a popular chip in the IoT world these days, not least of all because of it’s cheapness (approx $5 USD), its small size, its built-in wifi capabilities, and its relative power for its small size.

In my last tutorial, I wrote about how to use Alexa and an ESP8266 (with attached IR transmitter) to allow us to turn our TV off and on by voice command. That tutorial required, among other things, for the code running on the ESP8266 chip to connect itself to the local wifi network. In order to allow it to connect to our wifi, we hard-coded the wifi network ssid and password on the chip. Each of us who went through that tutorial had to customize those settings for our own local networks.

Now, while that was all well and good for a personal experiment in our homes, it’s obviously not at all suitable for a production-ready solution. Imagine that our device is to become an actual consumer product. We could hardly expect each customer to install Arduino or RTOS, prepare their development environments, set their wifi credentials in a .h file, and then compile & flash the code to the chip. Quite clearly we need to provide an easy user-friendly way to set the wifi credentials. And that must be the first thing that the user does after hooking up the device to a power supply; because without wifi credentials, the rest of the program won’t work.

Probably the easiest way to allow the user to control the wifi credentials (and maybe any other settings that we want to expose to the user as configurables) on the chip would be to allow the user to somehow discover and connect to the chip using their mobile devices. Easy, so we just run a server on the chip that accepts http requests from a client… oh no, wait. How will they know the IP? Ok so we run a UDP listener on the server, and then… oh, we need to be on the wifi for that. So we just… um… no wait… hard-code the wifi credentials?

We quickly see the conundrum we’re in. Logically, we need to either make the ESP8266 connect to something constant and reliable already on the network from which to read the config, or we need to make the ESP8266 a constant findable entity, and push the config to it. The first option is not practical. There’s a way, however, to implement the second way.

As a bonus, we’ll also introduce the use of an EEPROM database for on-chip storage of data. In the case of this tutorial, the data we’ll store is the wifi ssid and password.