



Workshop: iOS App with ESP8266 and Xcode

About Makerden

We're an Educational Makerspace where we combine the necessary tools, equipment, and instruction for project-based learning.

Members of our group vary both widely and wildly in terms of skill level and age range (10 to 68).



About Makerden

Events organized at our Educational Makerspace:

[DIY] These events include show-and-tells by our members followed by a work-on-your-project(s) and ask-for-help sessions.

[Workshop] An instructor leads a hands-on workshop for learning or polishing a practical skill useful for Makers (including schematic-drawing, PCB design, 3D-Printing, soldering).

[Class] An instructor leads a formal lecture featuring instructor-led activities that allow participants to learn, develop, or refine a technical skill.



About ACROBOTIC

Makerden's Hardware Electronics supplier

Open-Source electronics startup dedicated to the design of hardware and software products for use in education, DIY, hobby, arts, science, and more!

ESP-12E

Budget-friendly
Serial to Wi-Fi
Module



IOT Development Board
powered by the
ESP8266!

USB-ready

Program directly
from your
Arduino IDE!

Tutorials, product demos, projects: <http://learn.acrobotic.com>

Online store: <https://acrobotic.com>

Downloading this presentation

The screenshot shows a GitHub repository page for 'ESP8266_iOS_App'. The repository has 1 star and 0 forks. The 'Code' tab is selected. A prominent pink box highlights the URL https://github.com/makerdenio/ESP8266_iOS_App. The 'Download ZIP' button at the top right of the repository page is also highlighted with a pink border.

[Workshop][L1] Build an iOS App to Control LEDs from a smartphone using the ESP8266 Development Board! — Edit

https://github.com/makerdenio/ESP8266_iOS_App

Branch: master New pull request New file Find file HTTPS https://github.com/MakerdenIO/ESP8266_iOS_App Download ZIP

File	Description	Time
themakerbro Update README.md		Latest commit 0f3d825 13 minutes ago
esp8266_ios_app	first commit	5 minutes ago
presentation	first commit	5 minutes ago
LICENSE.txt	first commit	5 minutes ago
README.md	Update README.md	13 minutes ago

ESP8266_iOS_App

Build an iOS App to Control LEDs from a smartphone using the ESP8266 Development Board!

Intro

System overview

Software tools

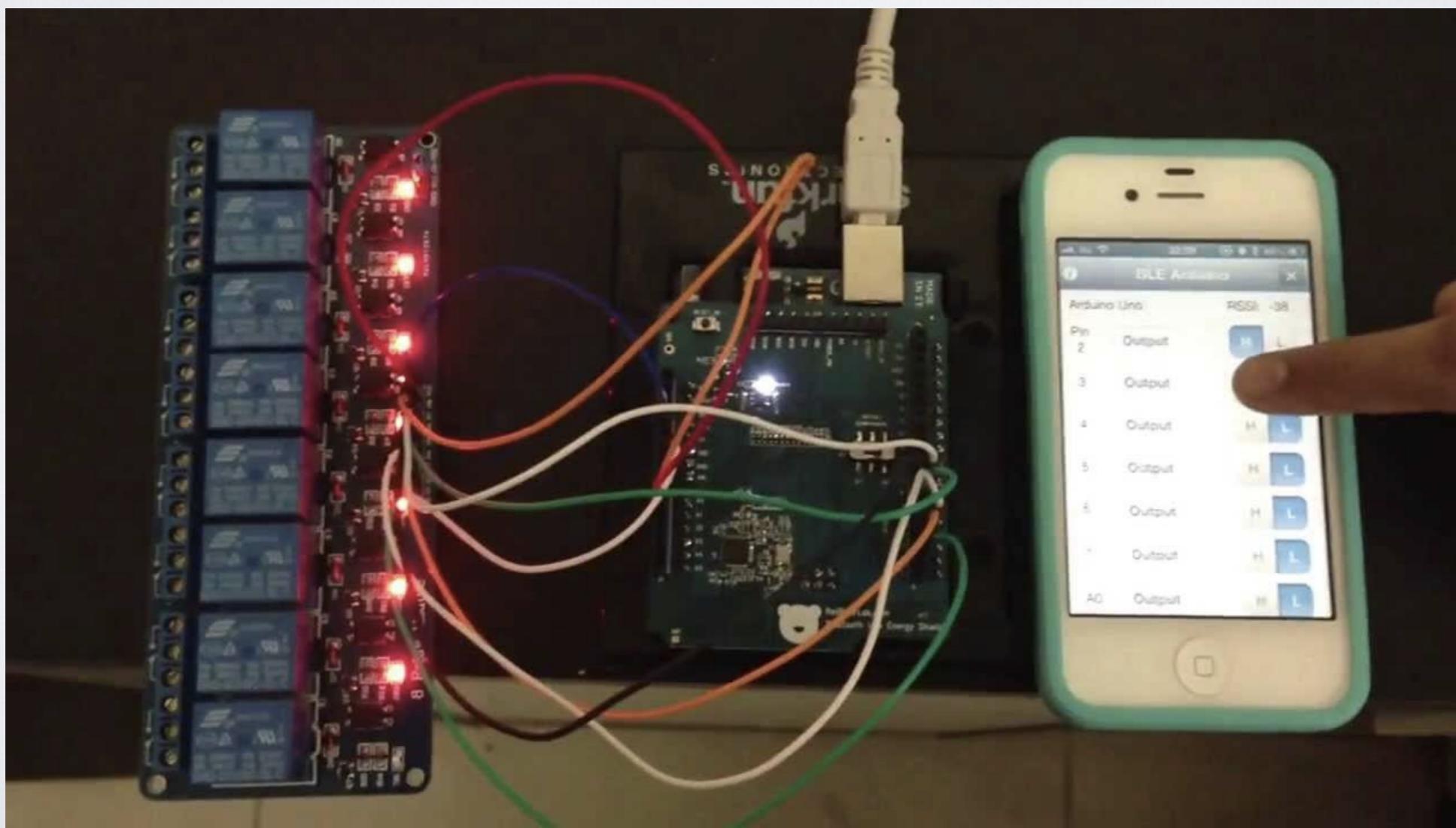
System Overview

Benefits of controlling Electronic Hardware from a smartphone

Offloads GUI to a separate device (< computation onboard)

Allows hardware to be controlled wirelessly

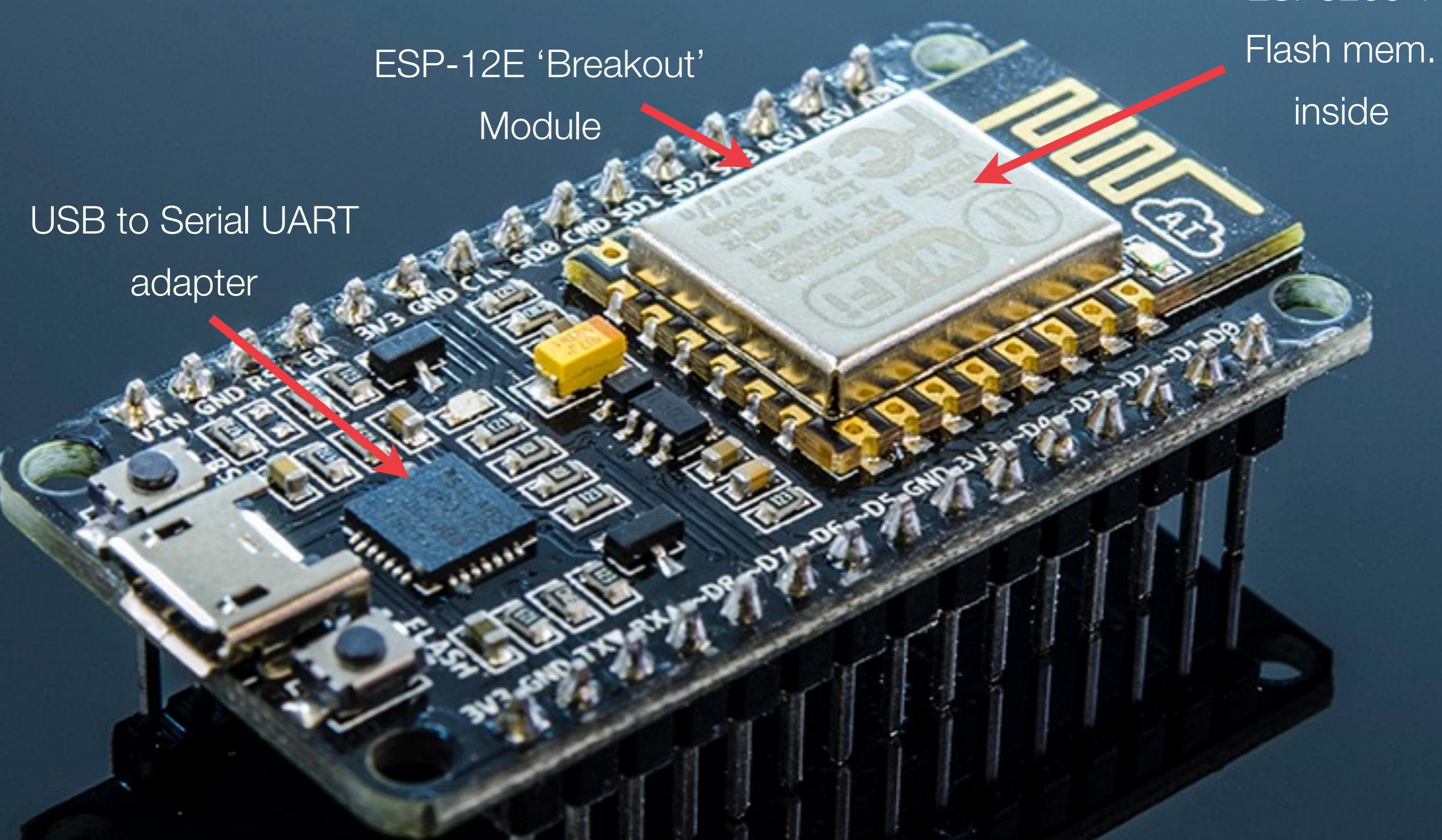
Allows hardware to be controlled remotely*



System Overview

The ESP8266 IC and Development Board

Serial to Wi-Fi adapter by Expressive Systems (mid-2014)

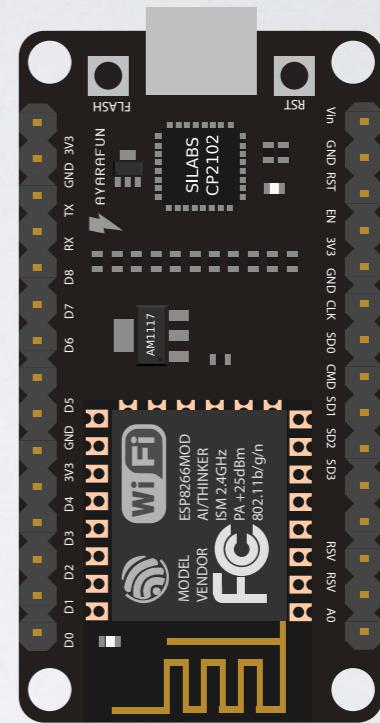
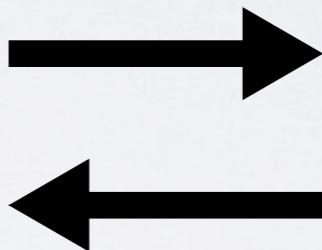


System Overview

The ESP8266 IC as a Soft Access Point



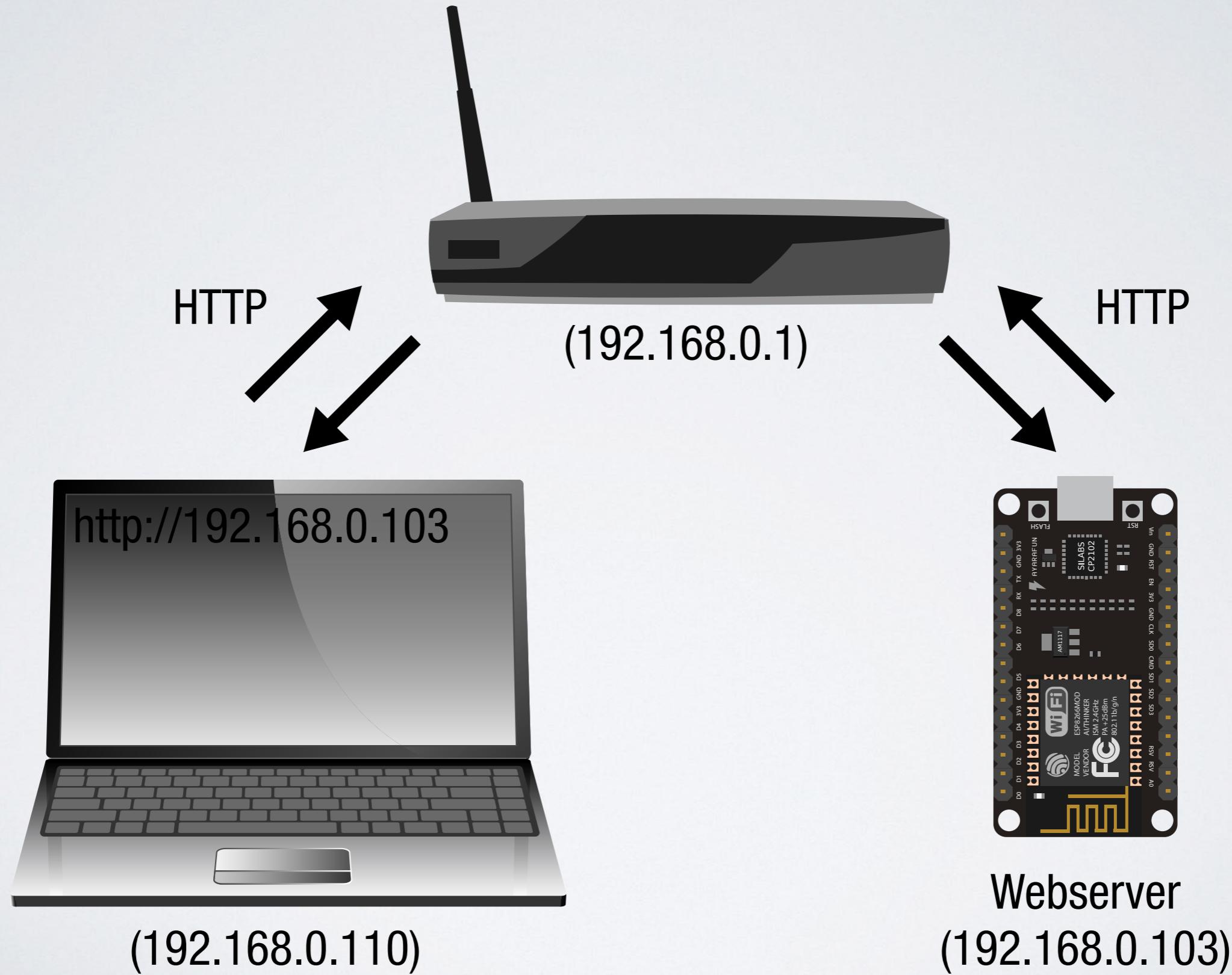
HTTP



Webserver (192.168.4.1)

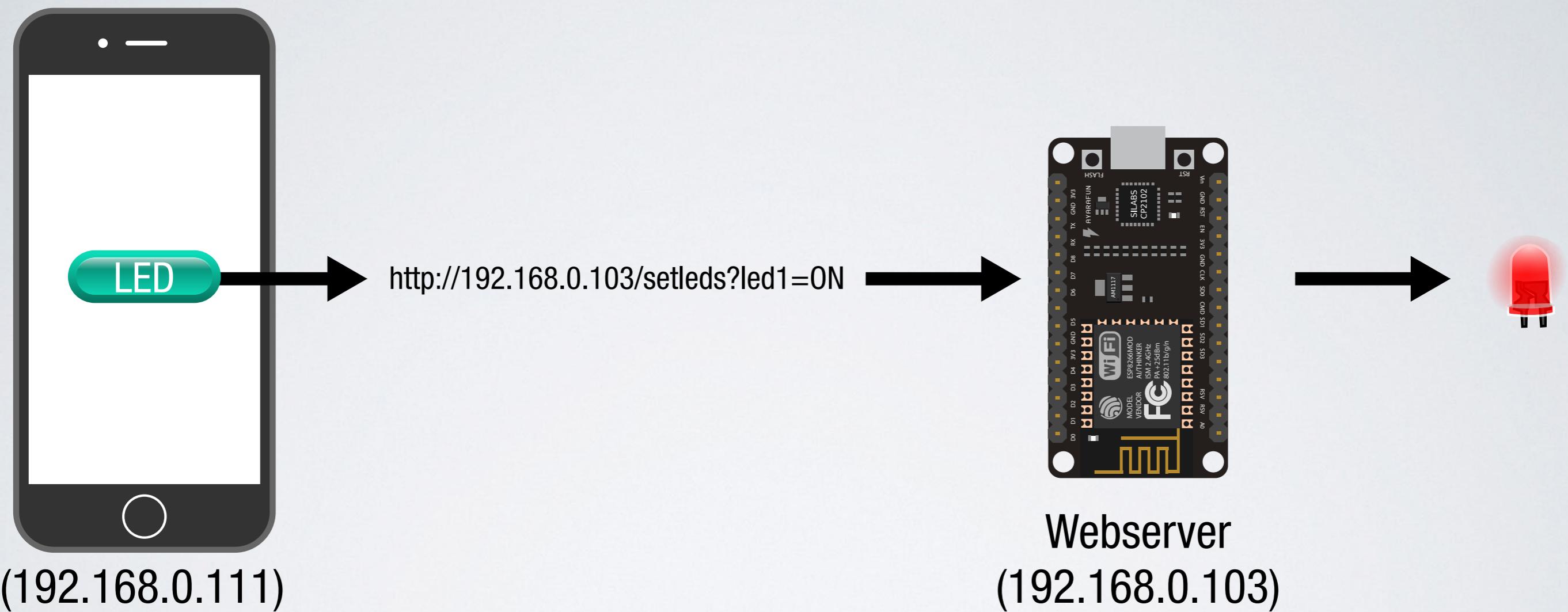
System Overview

The ESP8266 IC as a Station



System Overview

iOS App functional diagram

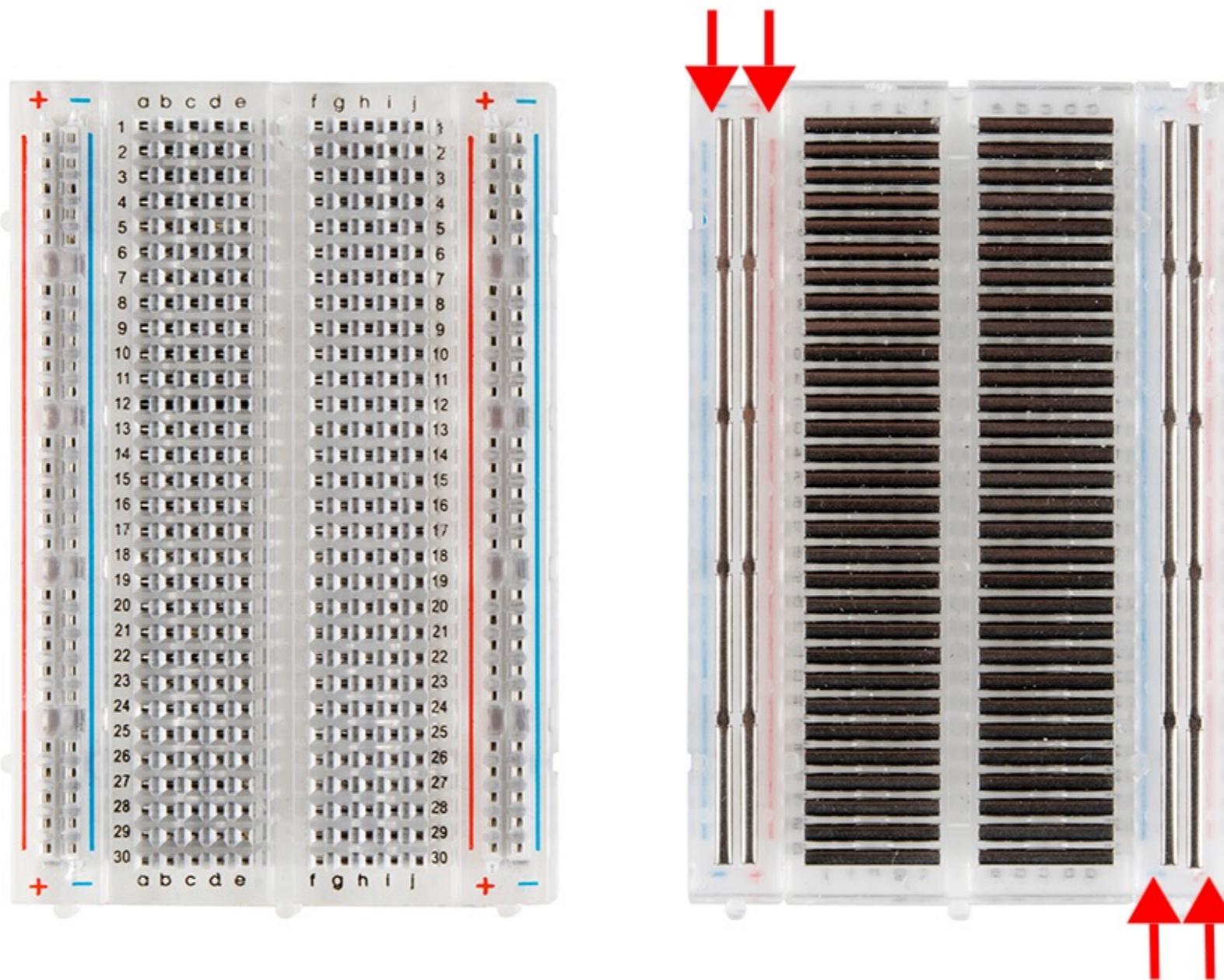


Fundamental Concepts

Hardware prototyping

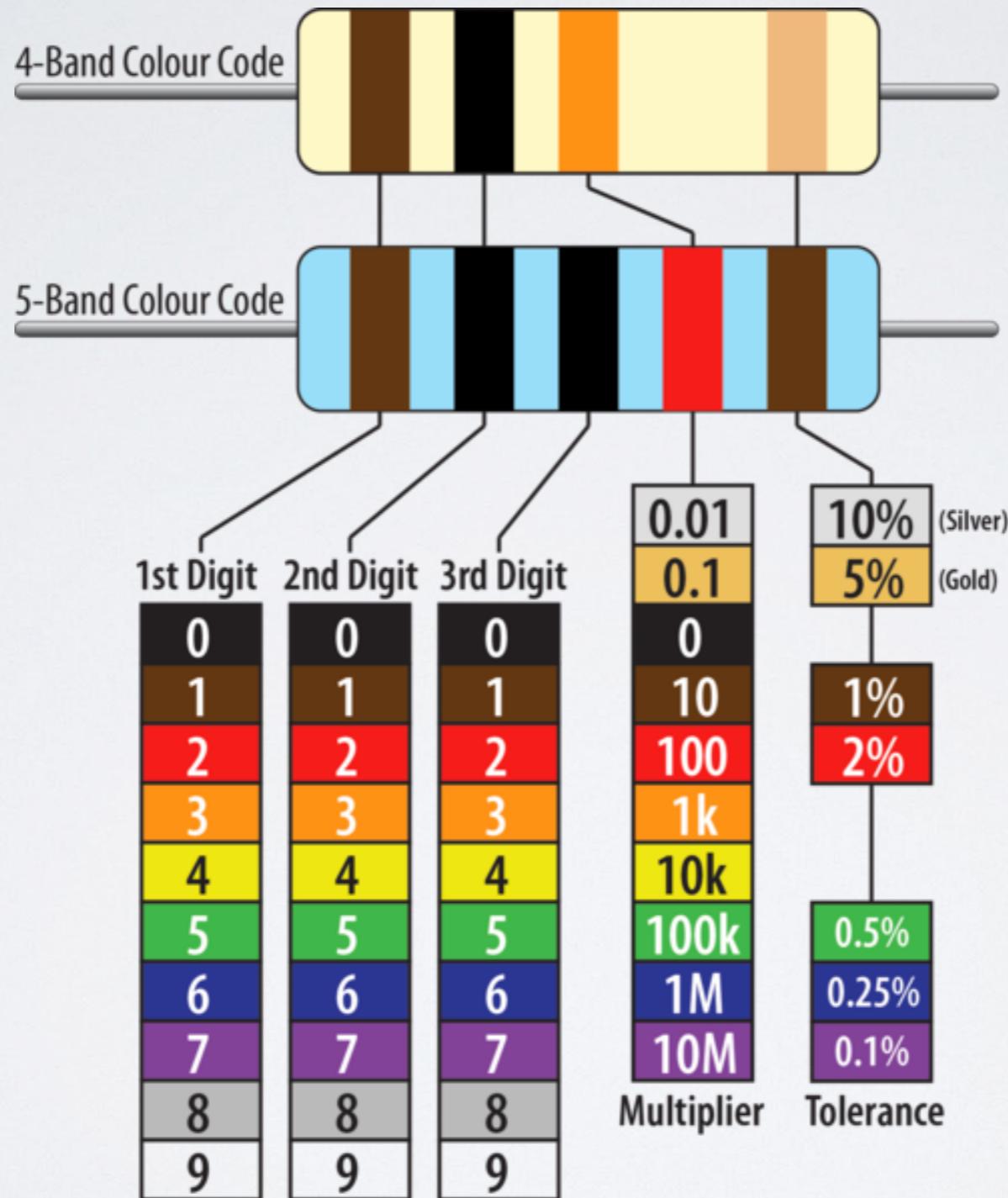
Electronic Fundamentals

How the solderless breadboard works:



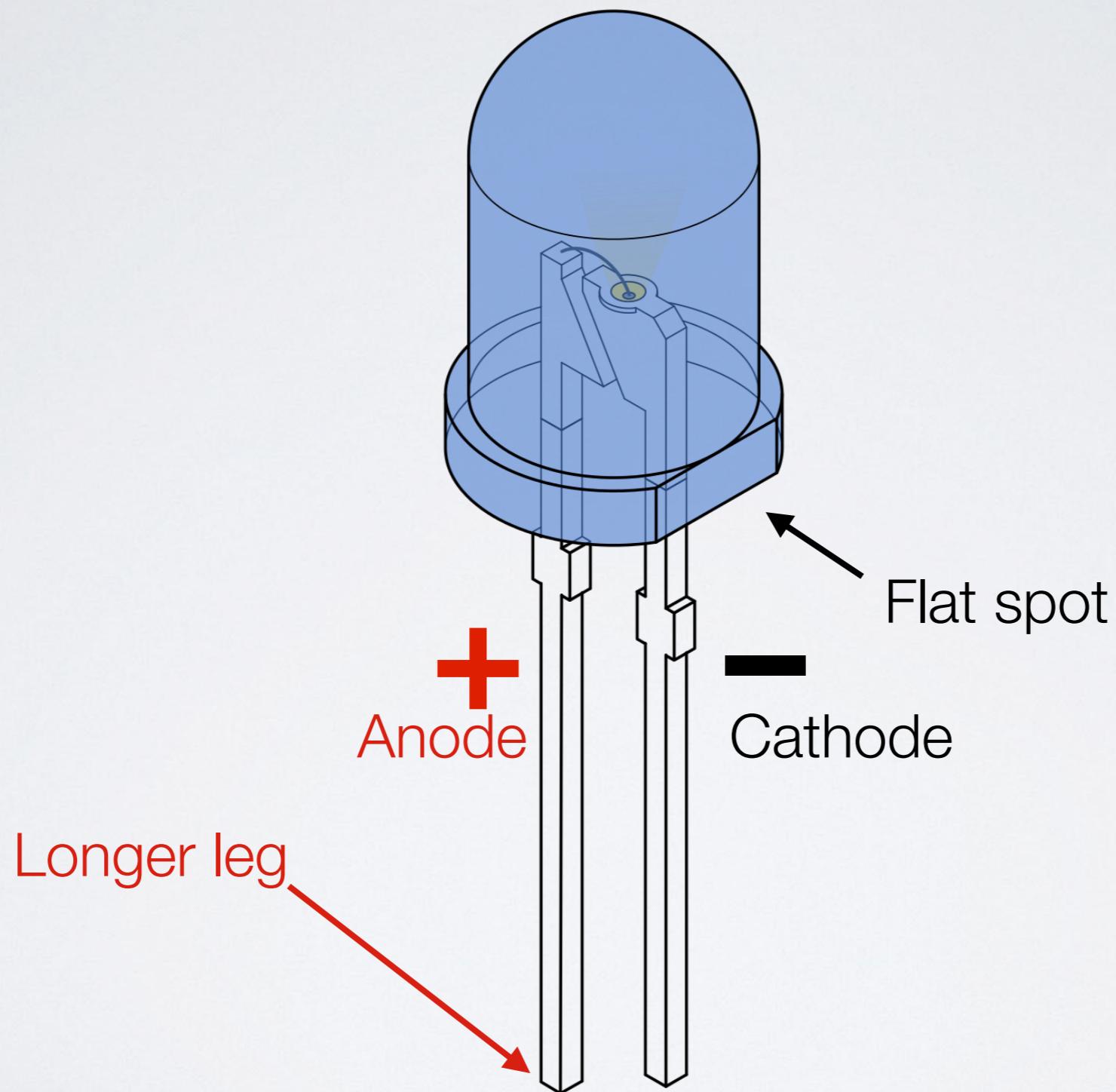
Electronic Fundamentals

Reading Resistor Codes:



Electronic Fundamentals

Reading LED polarity:

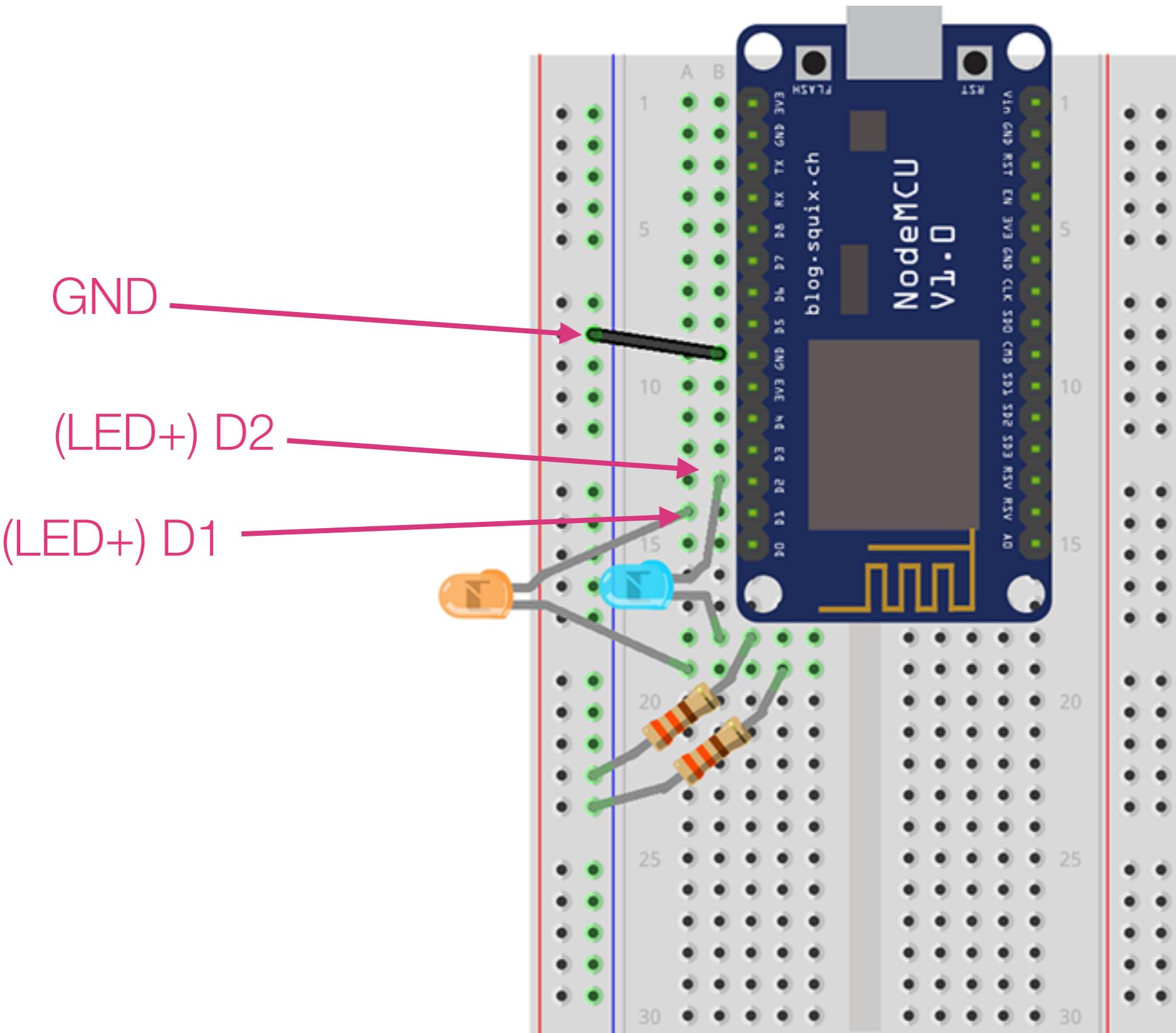


Setting Up the Hardware

Wiring two LEDs to GPIO on the ESP8266

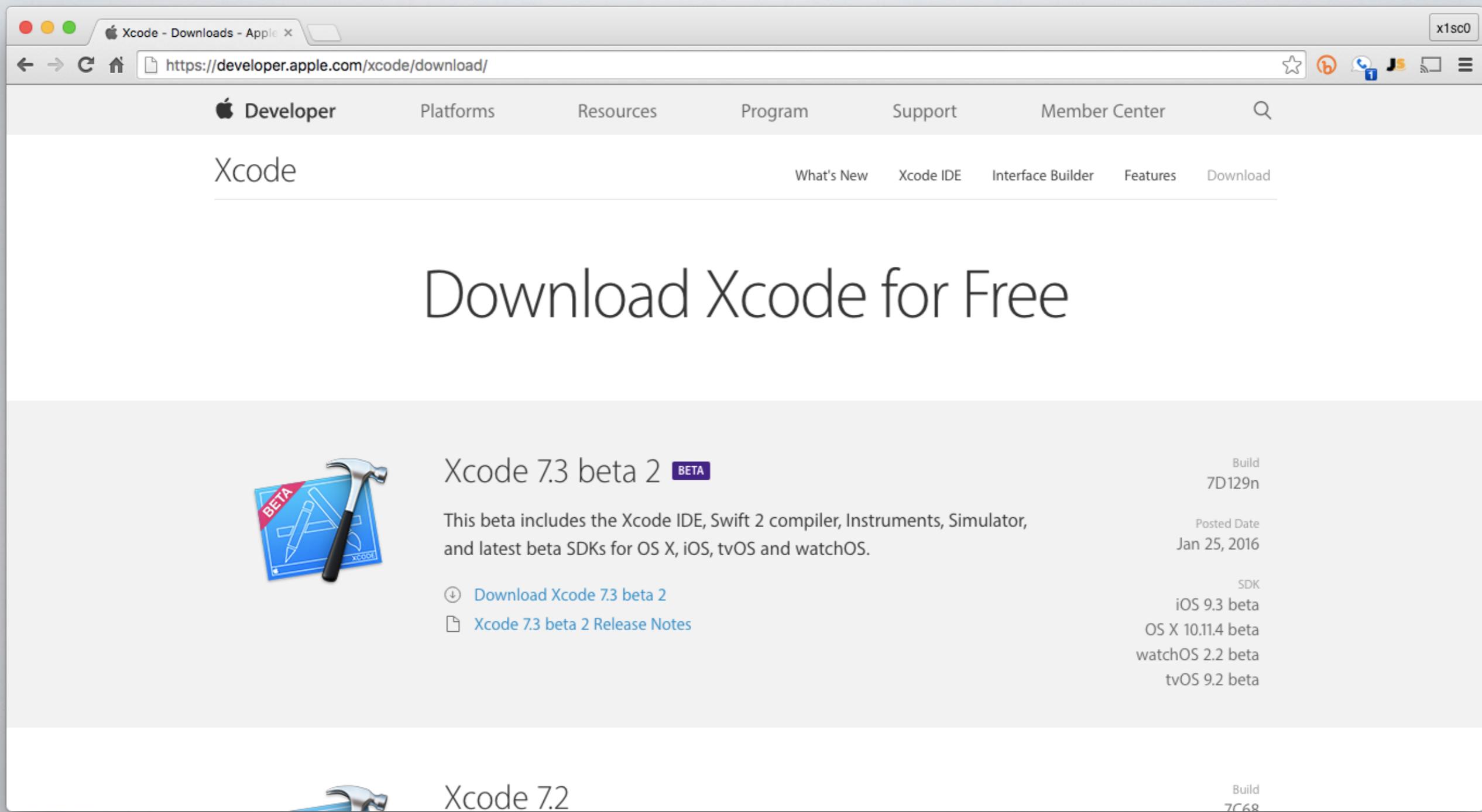
Setting Up the Hardware

Wiring up the circuit



Getting Started with Xcode 7

Installing Xcode 7—a computer application to develop apps for Apple devices.



The screenshot shows a web browser window with the URL <https://developer.apple.com/xcode/download/>. The page is titled "Xcode" and features a large heading "Download Xcode for Free". Below this, there are two main download options:

- Xcode 7.3 beta 2** (BETA) - Build 7D129n, Posted Date Jan 25, 2016. This beta includes the Xcode IDE, Swift 2 compiler, Instruments, Simulator, and latest beta SDKs for OS X, iOS, tvOS and watchOS. It is available for download and release notes are provided.
- Xcode 7.2** - Build 7C68. This version is also available for download and release notes are provided.

The page includes navigation links for Developer, Platforms, Resources, Program, Support, Member Center, and a search bar. The top right corner shows a user profile icon for "x1sc0".

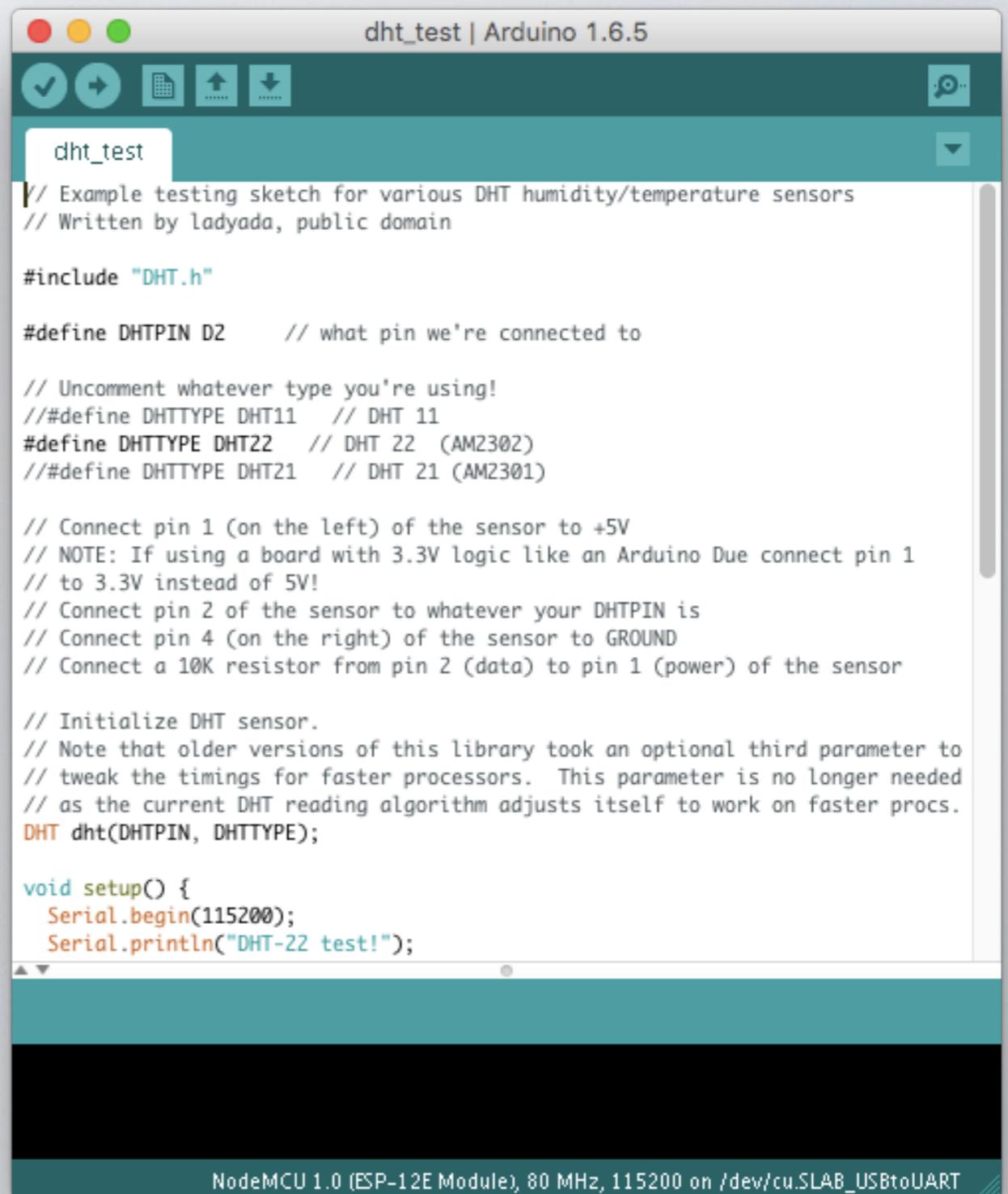
Getting Started with Arduino

Installing the Arduino IDE—a computer application to edit, compile, and upload our programs, as well as communicate via USB with the ESP8266 development board (and others)

Windows

Mac (OSX 10.5+)

Linux (32-bit, 64-bit)



Getting Started with the Arduino IDE

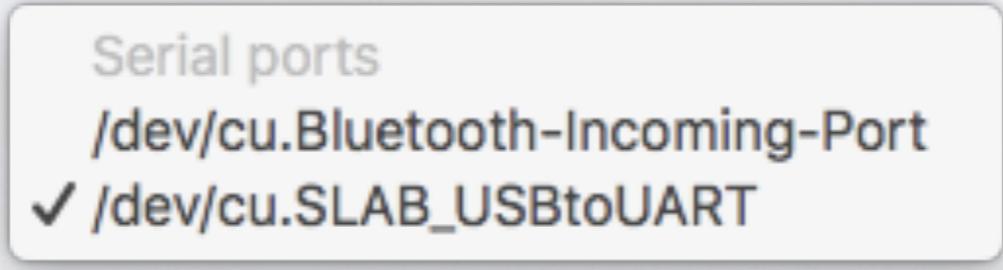
Configuring the Arduino IDE to support the ESP8266

Download and install the USB drivers:

<http://j.mp/ESP8266-driver>

In the Arduino IDE check under:

Tools > Port



Serial ports

/dev/cu.Bluetooth-Incoming-Port

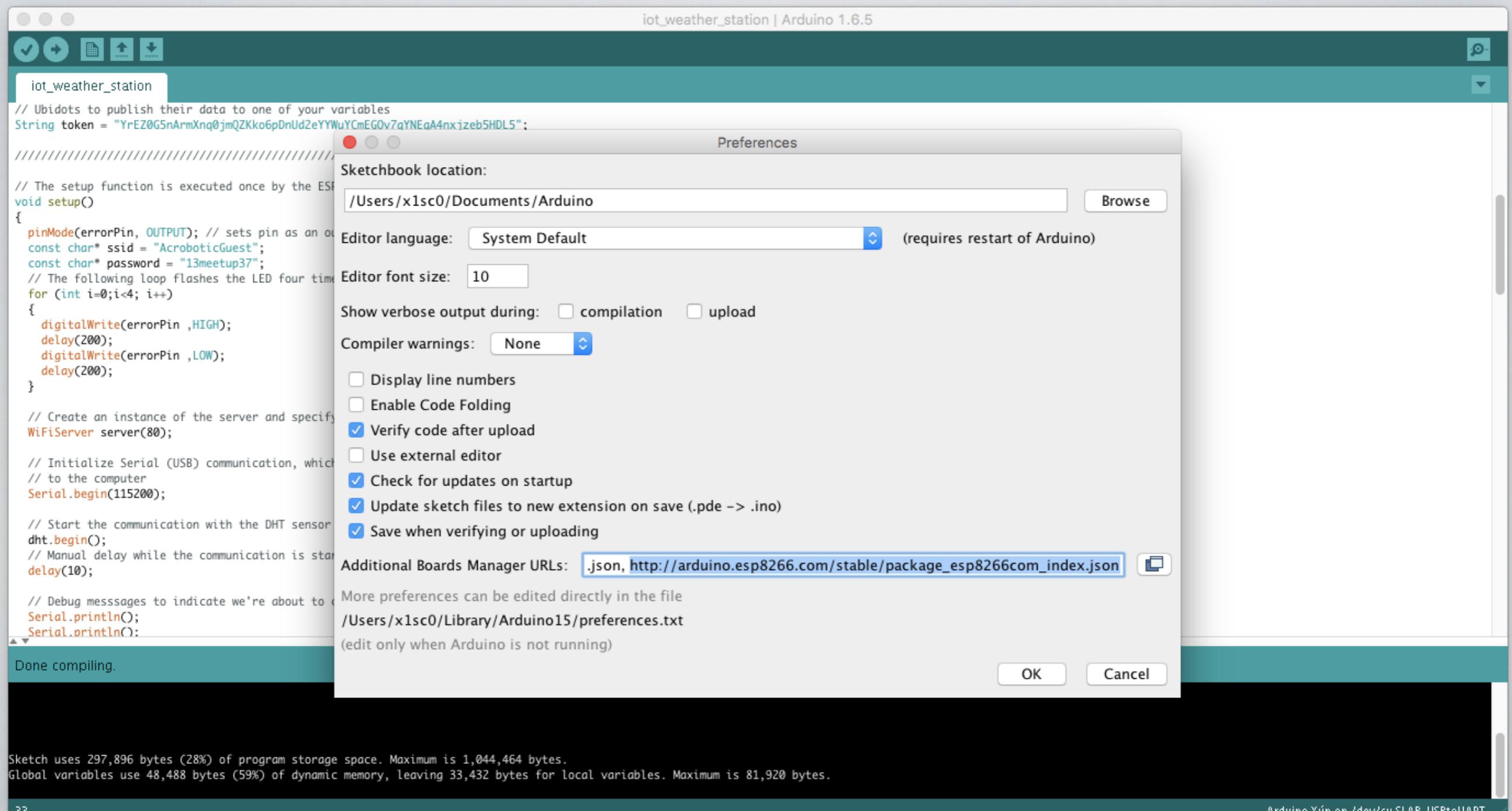
✓ /dev/cu.SLAB_USBtoUART

Getting Started with the Arduino IDE

Configuring the Arduino IDE to support the ESP8266

Navigate to “Preferences” and under Additional Board Manager URLs enter:

http://arduino.esp8266.com/stable/package_esp8266com_index.json



Sketch uses 297,896 bytes (28%) of program storage space. Maximum is 1,044,464 bytes.
Global variables use 48,488 bytes (59%) of dynamic memory, leaving 33,432 bytes for local variables. Maximum is 81,920 bytes.

Getting Started with the Arduino IDE

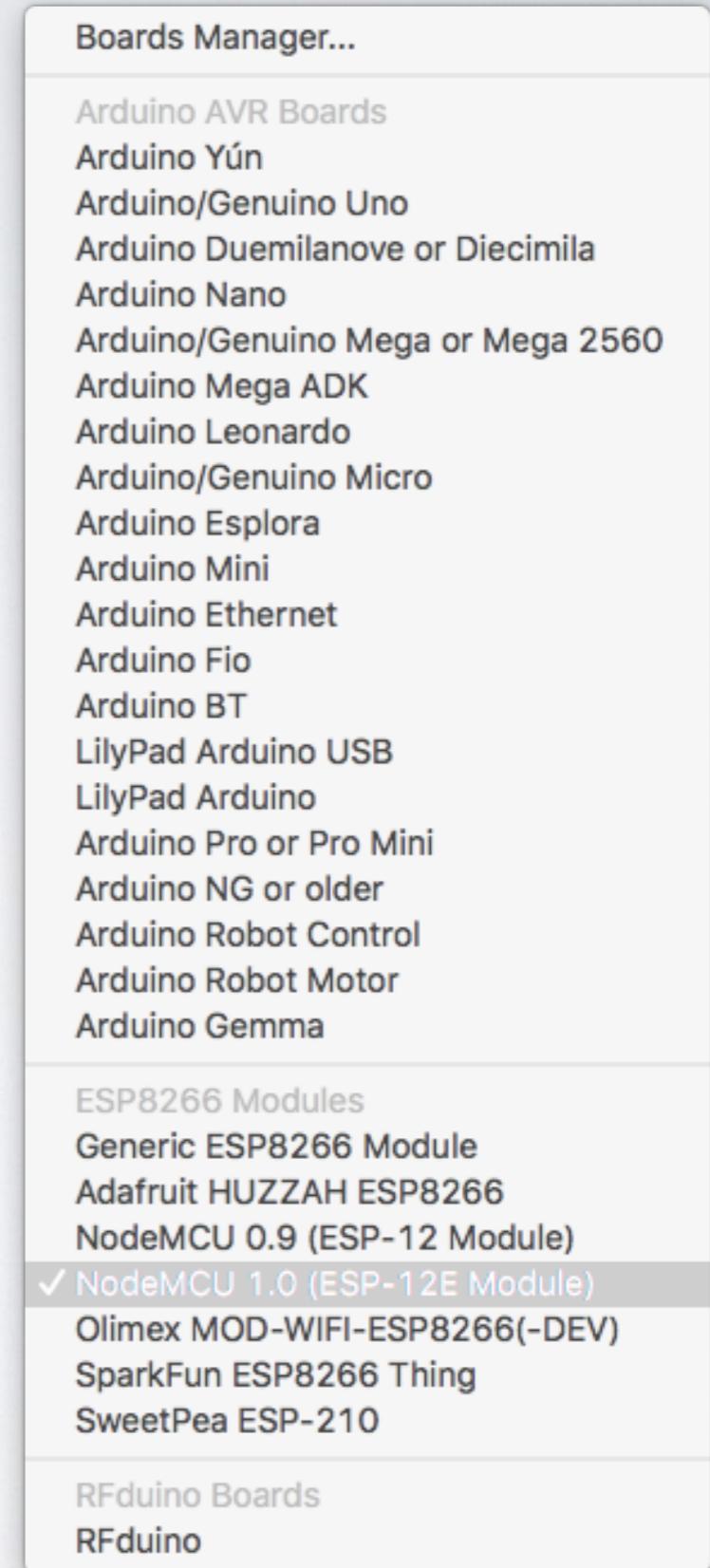
Configuring the Arduino IDE to support the ESP8266

Install the ESP8266 boards under:

Tools > Board > Boards Manager

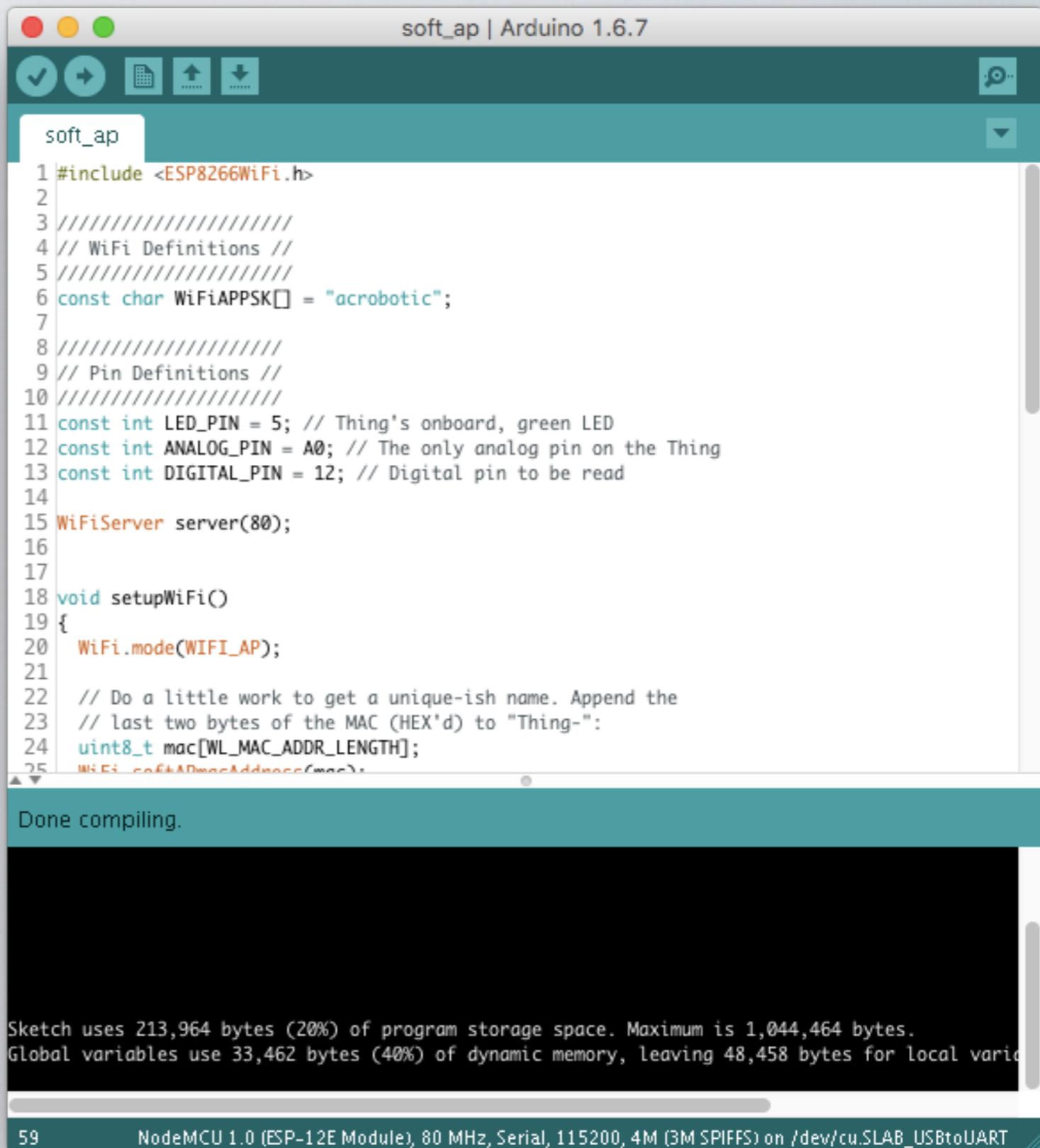
In the Arduino IDE select:

Tools > Board > NodeMCU 1.0 (ESP-12E)



Testing the Hardware

Upload **softap.ino** to the board
and check for the Wireless
Network “ACROBOTIC ESP8266
XXXX”



The screenshot shows the Arduino IDE interface with the title bar "soft_ap | Arduino 1.6.7". The main area displays the code for the "soft_ap" sketch. The code includes definitions for WiFi and pins, setup and loop functions, and a WiFiServer object. A message "Done compiling." is shown at the bottom. Below the IDE, a terminal window shows memory usage statistics.

```
soft_ap
1 #include <ESP8266WiFi.h>
2
3 /////////////////
4 // WiFi Definitions //
5 /////////////////
6 const char WiFiAPPSK[] = "acrobotic";
7
8 /////////////////
9 // Pin Definitions //
10 ///////////////
11 const int LED_PIN = 5; // Thing's onboard, green LED
12 const int ANALOG_PIN = A0; // The only analog pin on the Thing
13 const int DIGITAL_PIN = 12; // Digital pin to be read
14
15 WiFiServer server(80);
16
17
18 void setupWiFi()
19 {
20   WiFi.mode(WIFI_AP);
21
22   // Do a little work to get a unique-ish name. Append the
23   // last two bytes of the MAC (HEX'd) to "Thing-":
24   uint8_t mac[WL_MAC_ADDR_LENGTH];
25   WiFi.softAPmacAddress(mac);
```

Done compiling.

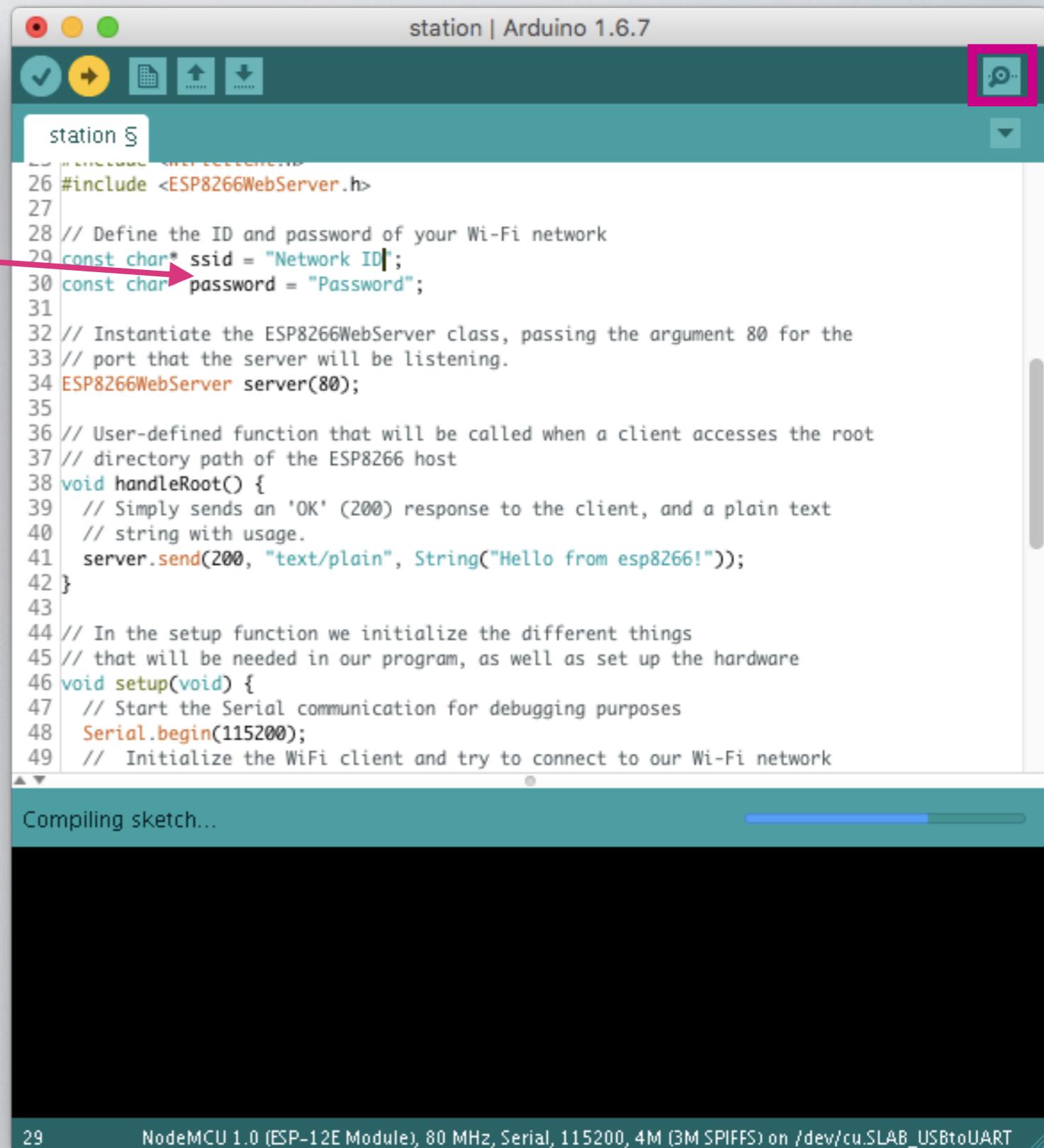
Sketch uses 213,964 bytes (20%) of program storage space. Maximum is 1,044,464 bytes.
Global variables use 33,462 bytes (40%) of dynamic memory, leaving 48,458 bytes for local varia

59 NodeMCU 1.0 (ESP-12E Module), 80 MHz, Serial, 115200, 4M (3M SPIFFS) on /dev/cu.SLAB_USBtoUART

Testing the Hardware

Change the SSID and password
to match your setup!

Upload **station.no** to the board
and open the Serial Monitor!



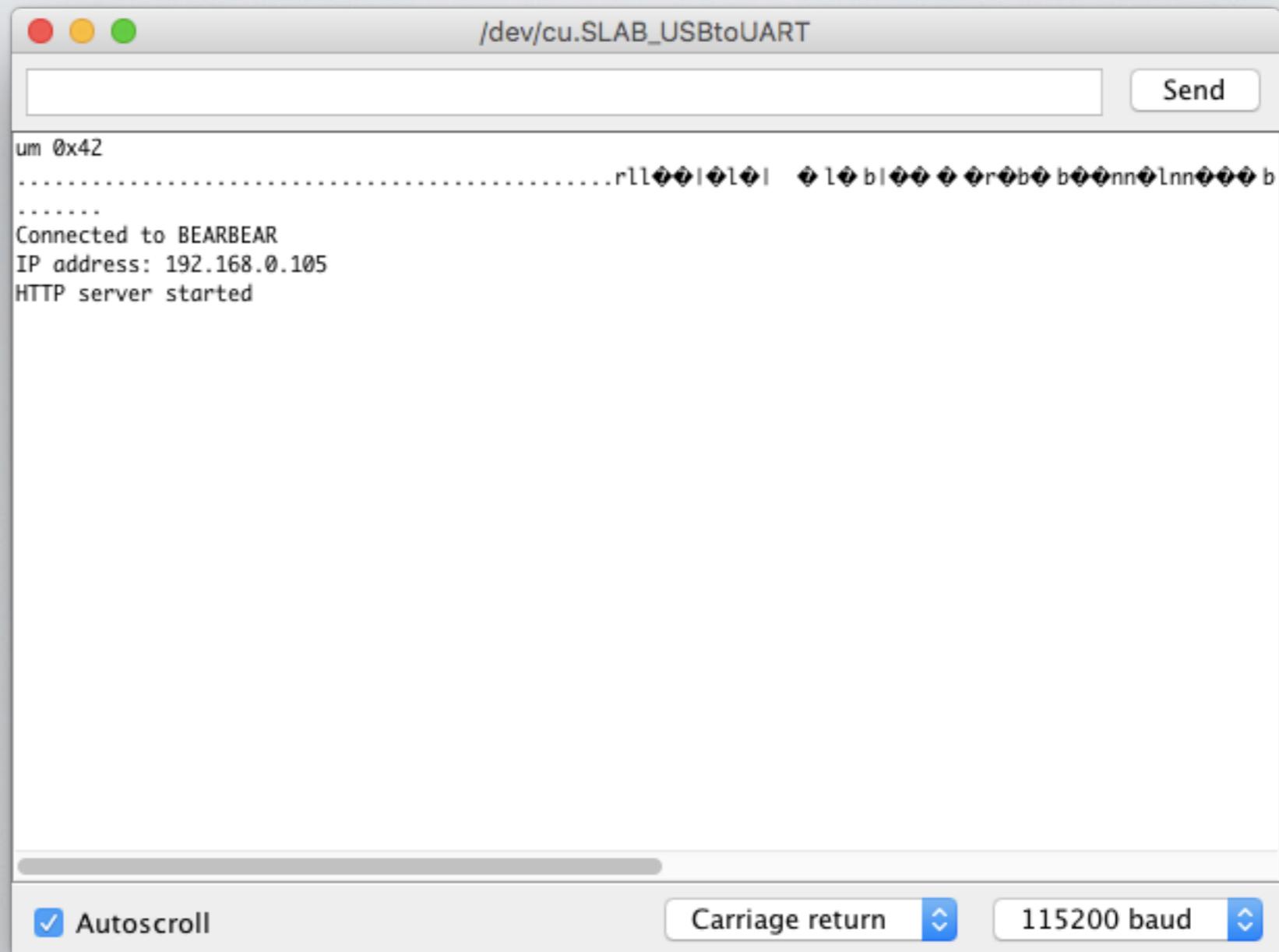
```
station | Arduino 1.6.7
station.ino
26 #include <ESP8266WebServer.h>
27
28 // Define the ID and password of your Wi-Fi network
29 const char* ssid = "Network ID";
30 const char* password = "Password";
31
32 // Instantiate the ESP8266WebServer class, passing the argument 80 for the
33 // port that the server will be listening.
34 ESP8266WebServer server(80);
35
36 // User-defined function that will be called when a client accesses the root
37 // directory path of the ESP8266 host
38 void handleRoot() {
39     // Simply sends an 'OK' (200) response to the client, and a plain text
40     // string with usage.
41     server.send(200, "text/plain", String("Hello from esp8266!"));
42 }
43
44 // In the setup function we initialize the different things
45 // that will be needed in our program, as well as set up the hardware
46 void setup(void) {
47     // Start the Serial communication for debugging purposes
48     Serial.begin(115200);
49     // Initialize the WiFi client and try to connect to our Wi-Fi network
```

Compiling sketch...

29 NodeMCU 1.0 (ESP-12E Module), 80 MHz, Serial, 115200, 4M (3M SPIFFS) on /dev/cu.SLAB_USBtoUART

Testing the Hardware

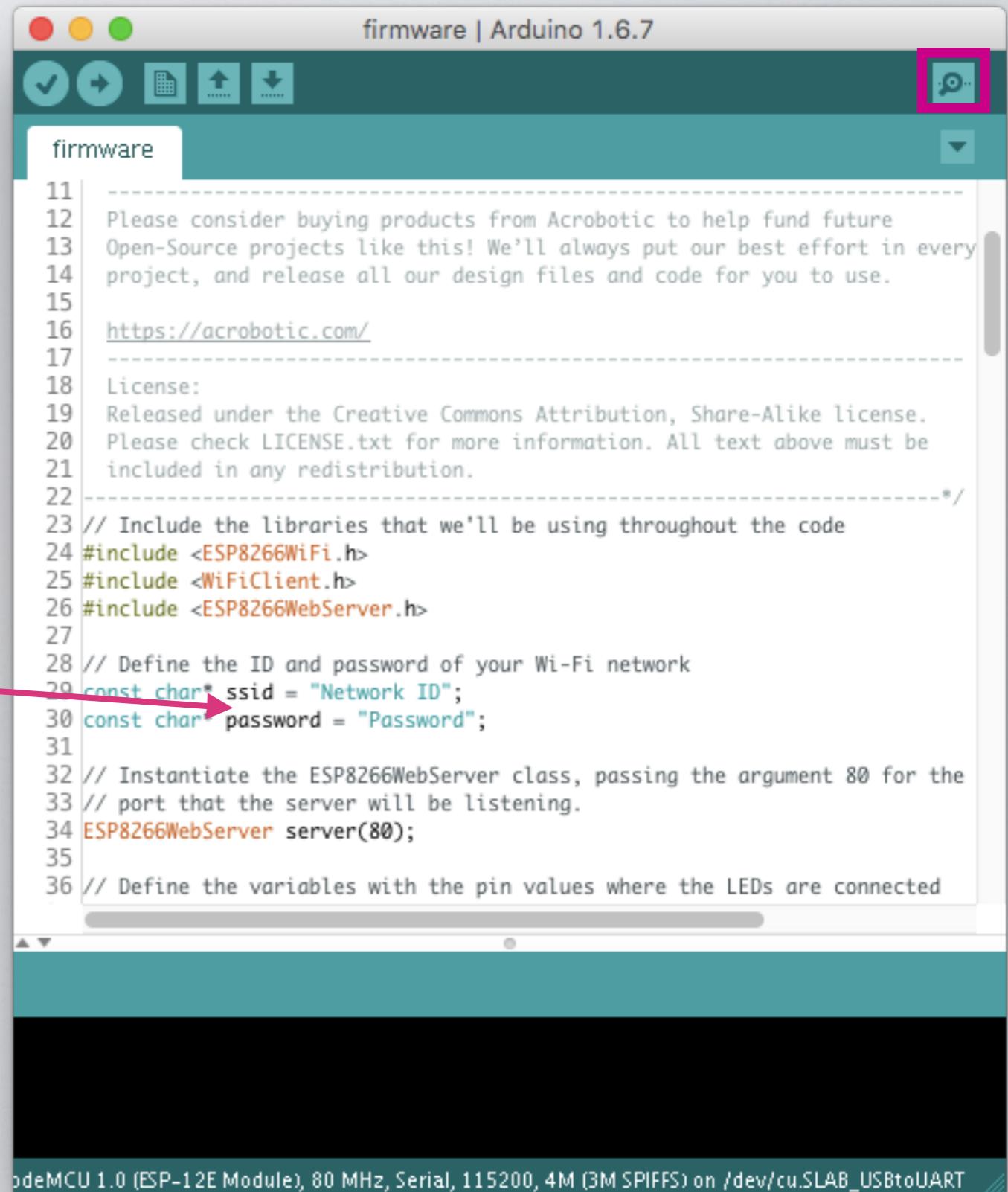
Upon a success fun connection,
you should see:



Loading the Project Firmware

Upload **firmware.ino** to the board and open the Serial Monitor!

Change the SSID and password to match your setup!



```
firmware | Arduino 1.6.7

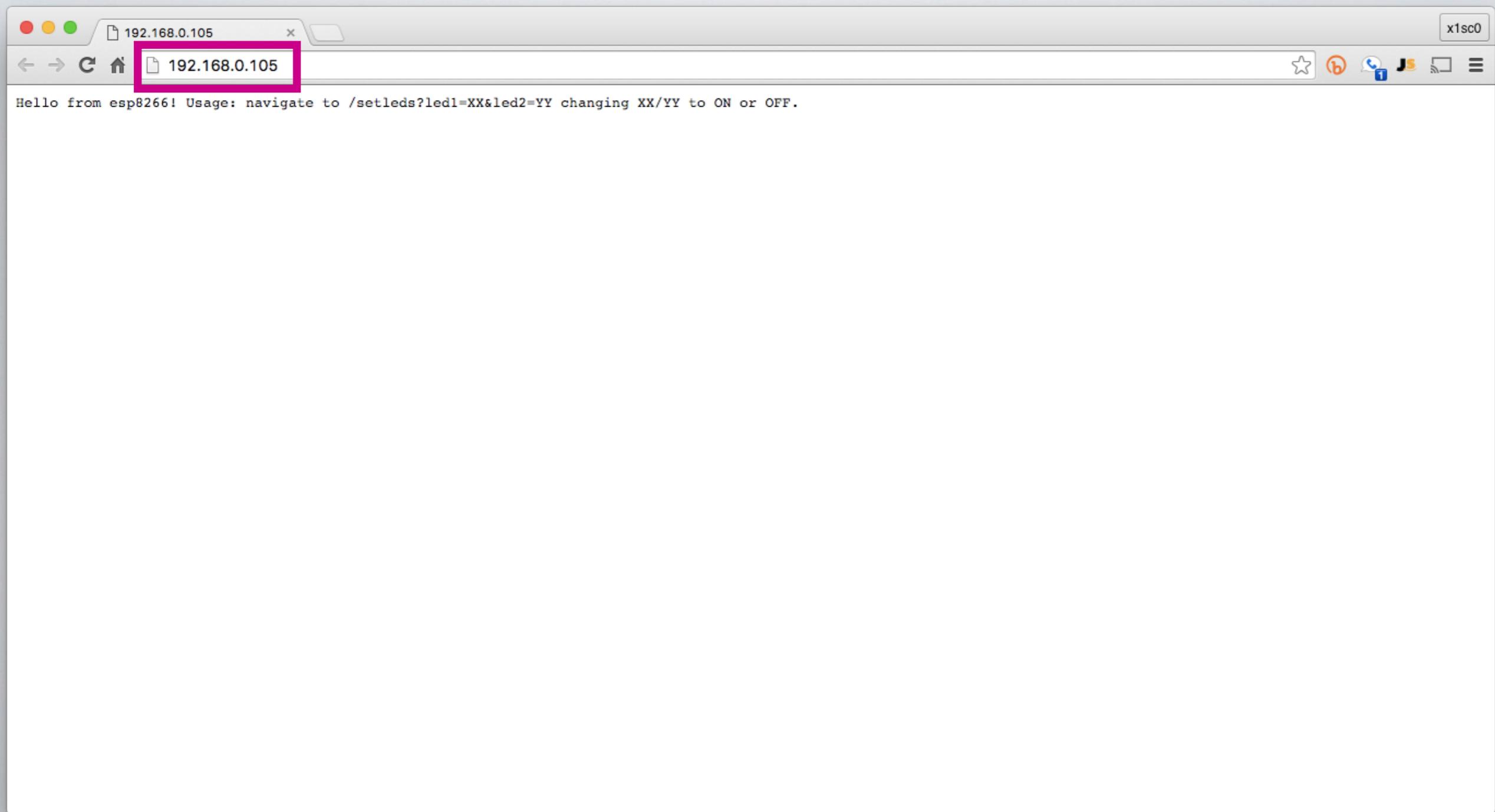
firmware

11
12 Please consider buying products from Acrobotic to help fund future
13 Open-Source projects like this! We'll always put our best effort in every
14 project, and release all our design files and code for you to use.
15
16 https://acrobotic.com/
17
18 License:
19 Released under the Creative Commons Attribution, Share-Alike license.
20 Please check LICENSE.txt for more information. All text above must be
21 included in any redistribution.
22 */
23 // Include the libraries that we'll be using throughout the code
24 #include <ESP8266WiFi.h>
25 #include <WiFiClient.h>
26 #include <ESP8266WebServer.h>
27
28 // Define the ID and password of your Wi-Fi network
29 const char* ssid = "Network ID";
30 const char* password = "Password";
31
32 // Instantiate the ESP8266WebServer class, passing the argument 80 for the
33 // port that the server will be listening.
34 ESP8266WebServer server(80);
35
36 // Define the variables with the pin values where the LEDs are connected
```

CodeMCU 1.0 (ESP-12E Module), 80 MHz, Serial, 115200, 4M (3M SPIFFS) on /dev/cu.SLAB_USBtoUART

Testing the Project Firmware

Use your browser (laptop or mobile) and navigate to the IP address of the ESP8266!



Setting Up the App

Using Xcode 7 to develop a Control App