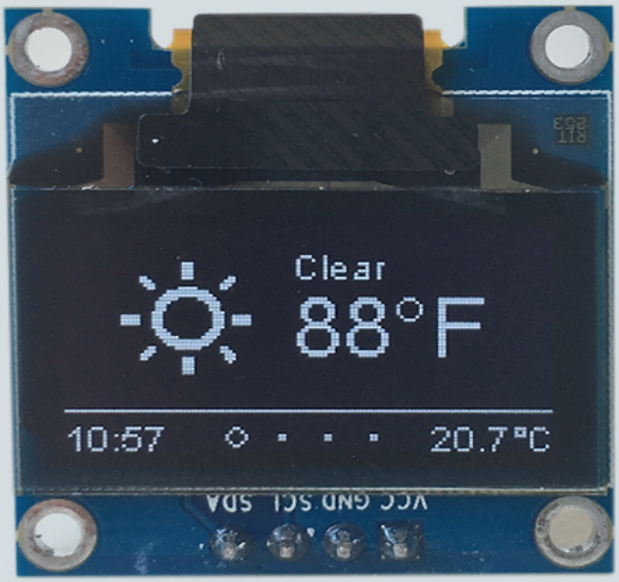
Introduction to
Internet of Things

ESP8266
WEATHER STATION

GETTING STARTED GUIDE



DANIEL EICHHORN

ESP8266 Weather Station

Getting Started Guide

Daniel Eichhorn

This book is for sale at http://leanpub.com/esp8266weatherstationgettingstartedguide

This version was published on 2018-09-11

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Introduction

Since the end of 2014 the ESP8266 chip by Chinese manufacturer Espressif has gained a lot of popularity in the DIY community, due to its rich set of features but also due to the very attractive price. First it was only available as a WiFi extension to existing development boards, cutting the price of comparable products from USD $60 to a mere $6! Suddenly all the Arduino developers had an affordable way to connect their devices to the internet. Not long after, clever hackers and engineers realized that the ESP8266 could be used beyond the rather simple AT firmware. A software development kit (SDK) was available but badly documented, so they reverse-engineered the SDK and used Google Translate to understand the Chinese manual.

At first the process to set up a development environment was complicated and cumbersome. Files had to be downloaded from different sources and copied to various locations. But then several groups started to provide simplifications to this process. One of the first simplifications was the NodeMCU Lua firmware which could interpret scripts written in the language Lua at runtime. The firmware also provided bindings into Espressif's API from the Lua language so that the pins of the ESP8266 could be easily controlled with just a few lines of code.

A few months later another huge simplification became available: the integration of the C/C++ API into the Arduino IDE! Suddenly it was possible to profit from the simplicity of the Arduino ecosystem, which not only provided a vast number of libraries but also made the C programming start of your project a lot easier. Since code developed in the Arduino IDE compiled into a very efficient binary the often scarce resources of the ESP8266 were also used more efficiently. For instance, the interpreter (the program that reads and executes scripts) of the Lua firmware needed a lot of memory just for itself and did not leave much for your script code.

After having used the Lua firmware for a while I got frustrated by its instability and lack of peripheral support. So I just jumped on the possibility to program the ESP8266 from the Arduino IDE - and I loved it from the beginning. I didn't have to worry about a complicated tool installation: it was as simple as copying a URL into the right spot in the Arduino IDE. And also many libraries programmed for the standard Arduino ATmega chips worked out of the box for the ESP8266 as well! So I went to work and ported some of the projects I had written for the Lua firmware to the Arduino/ESP8266 platform.

However, I was struggling in Lua with one peripheral module I already had successfully working: a wonderfully crisp OLED display. There were several libraries available for the Arduino using that display but I just couldn't get them to run: the extremely versatile and rich u8glib used a lot of ATmega specific code and just wouldn't compile. The Adafruit library on the other hand was made for slightly different displays and wouldn't work for me either. So I set out and started to write my own (and very first) library for the Arduino/ESP8266 platform.

To verify the library I implemented a few ideas which involved the OLED display. One of them was the ESP8266 WeatherStation. After getting the code to work I wrote a blog post about it and had

it running somewhere in my apartment - and I forgot about it until I saw that suddenly the visits on that blog post spiked and that many visitors came from Thingiverse. From a 3D printing project built around my WeatherStation code, that was the moment when I realized that I had something interesting and people had found the WeatherStation appealing.

I decided to provide the right components needed for building the WeatherStation and to sell it as a kit for the ESP8266 WeatherStation. Quickly I had set up a simple PayPal shop on my blog. A supplier in China would ship the kit directly to buyers all over the world and after a few months WeatherStations were being programmed in more than 20 countries.

You are now holding a guide to the WeatherStation in your hands. Thank you for your interest! You might have just found this guide on Amazon and you don't have the hardware yet. Or you have already acquired the components on your own and are now looking for a guide to use them. Or you have bought the kit from my shop or my listing on Amazon. In all of these cases you quickly want to get started with the ESP8266 and I've tried very hard to make this as easy as possible for you. Please let me know if you see mistakes. You can reach out to me through dani.eichhorn@squix.ch or on Twitter: https://twitter.com/squix78

Also make sure that you subscribe to my newsletter to stay updated with latest news around the ESP8266. You will get a maximum of 1-2 emails per month, I promise! https://blog.squix.org/subscribe

One more thing! If you like this project please have a look at my shop. I recently created a hardware kit containing a beautiful color display with touch screen. There are are several projects you can build with it.

The ESP8266 WiFi Color Display Kit

https://thingpulse.com/product/esp8266-wifi-color-display-kit-2-4/

Required Hardware

The Starter Kit is available from two shops. You can buy it from the shop on my blog and shipping is available to almost all destinations: https://thingpulse.com/product/esp8266-iot-electronics-starter-kit-weatherstation-planespotter-worldclock/

If you live in the US you can purchase the WeatherStation from Amazon (https://www.amazon.com/dp/B01KE7BA3O/) as well.

The ThingPulse ESP8266 WeatherStation Kit has the advantage that everything fits together, but you can of course also get the components from your preferred supplier. In this chapter I will quickly go through the minimal requirements and the options you have to build your first WeatherStation.

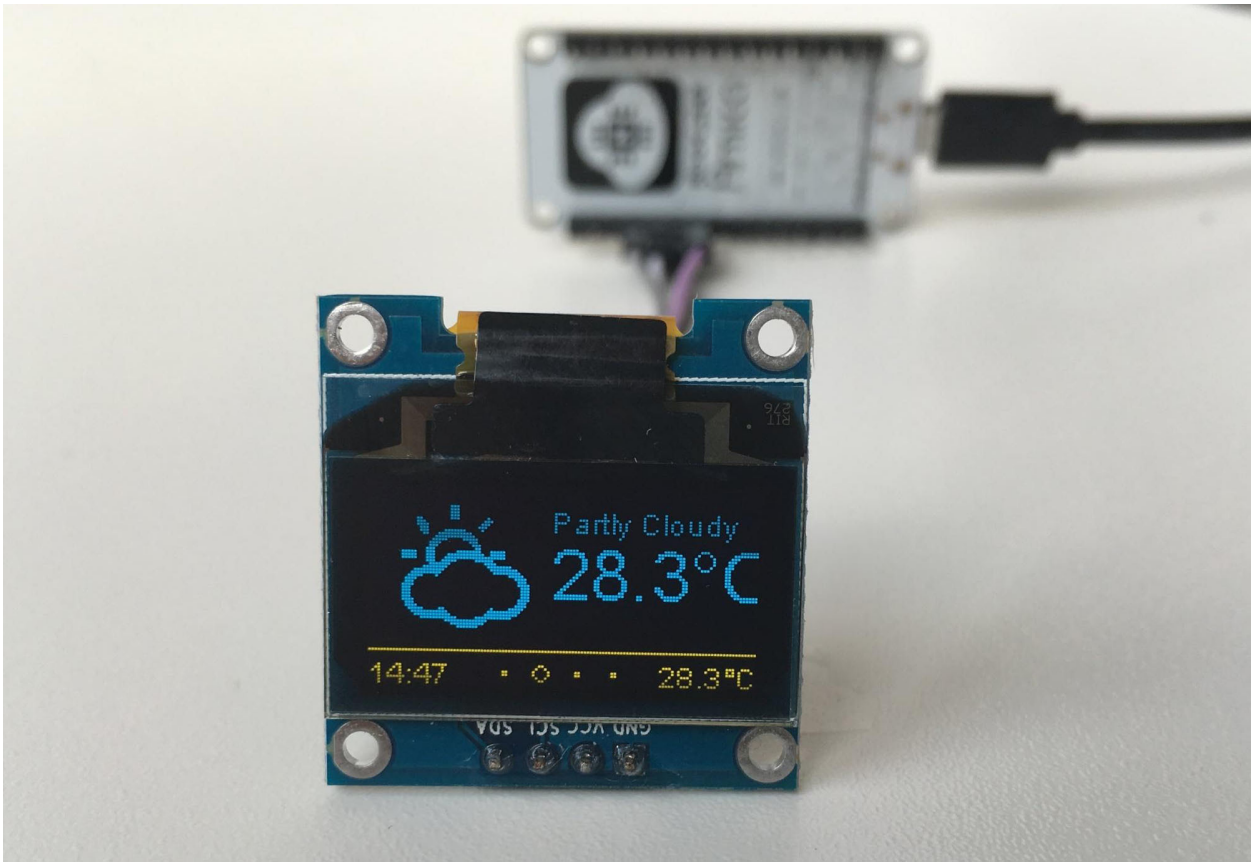
The ESP8266 Weather Station starter kit

ESP8266 Module

There are many different modules available based on ESP8266s; they differ in a number of aspects such as the quantity of available GPIO pins or if they can be programmed easily without need of an additional Serial-to-USB converter. If you are a beginner I suggest you use a developer-friendly module like the NodeMCU V1.0 or the Wemos D1 mini. They come with a USB connector and have the maximum number of available pins ready for your usage. The absolute minimal requirement is that your ESP8266 module has at least two free GPIO pins to connect it to the OLED display.

OLED Display

With the display you also have many options: do you want the pixels to be white or blue, or do you even prefer a two color display where the footer is in one color and the rest in another? What really matters is the driver chip and the protocol. The OLED library currently supports I2C and SPI for both the SSD1306 and the SH1106 chip. The first is often used for 0.96" inch displays while the second one is used for 1.3" displays. Displays with SPI interface will consume more of your free GPIO pins.



The OLED display with a blue and a yellow section

Wires & Cables

You will also need some wires to connect the display to the ESP8266. In case you want to connect the display directly to the NodeMCU you will need at least four female-to-female jumper wires, since both the display and the NodeMCU have male pin headers. The wires don't need to be long, 4" (10cm) is usually enough.

To program the ESP8266 module you will also need a USB cable. In case of the NodeMCU this cable

should have a micro-USB connector on the module side and a normal USB connector for your PC or Mac.

Tool Setup

In this chapter we will prepare your development environment by installing all the tools necessary. Drivers are needed to communicate with the ESP8266, a tool called “Arduino IDE” will let us write code, and a sample project will prove that the components are working well together.

The NodeMCU ESP8266 Module

Download and Install the Serial Driver

To program the NodeMCU V1.0, your development platform (PC, Mac, Linux) needs to detect the Serial-To-USB adapter soldered onto the ESP8266 module. There are two different versions: some have the CP2102 Serial-To-USB adapter; others have the CH340. My guess is that most new modules come with the CH340 chip.

If your module has the CP2102 converter then you can download and install the driver from here: https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx

In case your module comes with a CH340 serial-to-USB converter then download the drivers from here:

- Win: http://blog.squix.org/downloads/CH341SER.zip
- Mac: https://blog.squix.org/wp-content/uploads/2016/12/CH34x_Install_V1.3.zip

The Arduino IDE

The Arduino Integrated Development Environment (IDE) is the tool you will use to program the ESP8266. IDEs are more than just editors; they help you with various tasks during the development process. For me as a professional software developer the Arduino IDE is not a very powerful one. It lacks some features that I got used to and I am missing them every time I program for the ESP8266. But the Arduino IDE was not made for professional programmers, it was made with the beginner in mind and this is also the reason why we will use it here. If you are looking for more convenience, have a look at http://platformio.org/ or the ESP8266 integration into the Eclipse IDE.

To install the Arduino IDE go to https://www.arduino.cc/en/Main/Software and download the latest version matching your operating system:

- For Mac OS X you can download a ZIP file which you then have to extract. Take the extracted application “Arduino” and move it to your Applications folder.
- For Windows you have the option between an executable installer and a ZIP file. The ZIP file might be the better option if you do not have administrator permissions on your system. The installer on the other hand can put the libraries in the proper places.

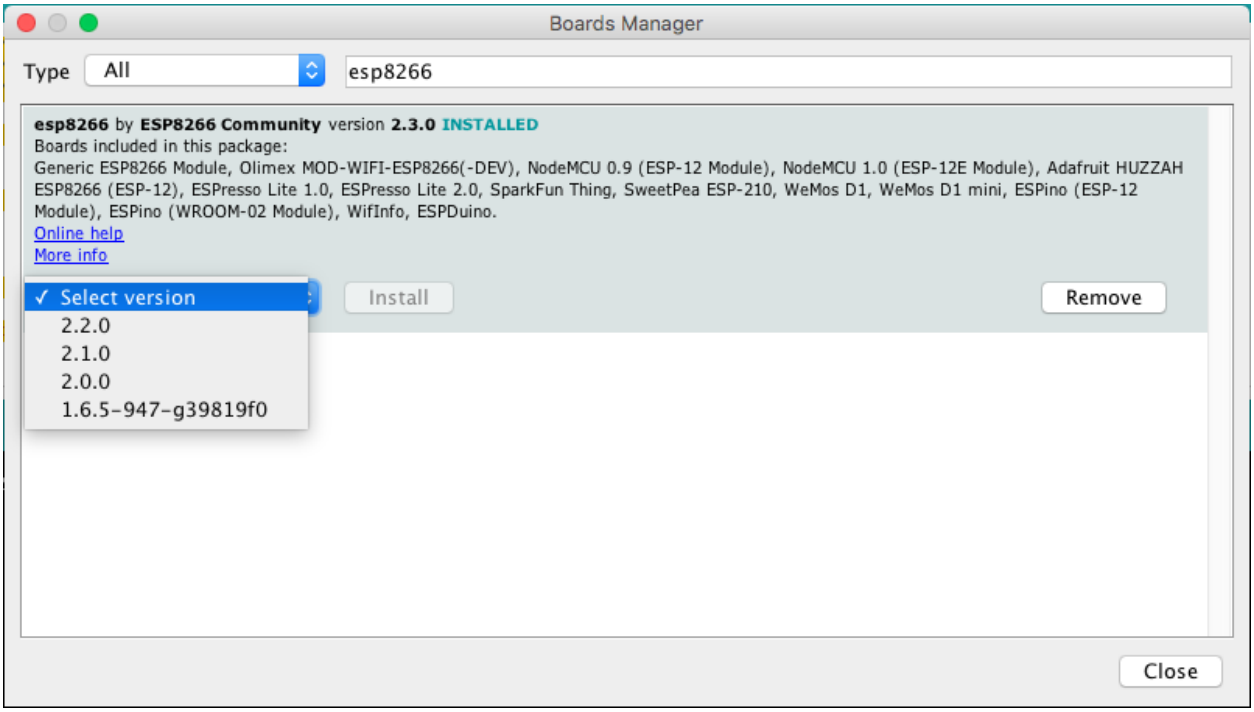
Now you have a bare Arduino IDE which brings everything needed to write programs for the standard Arduino ATmega chips. But we want to write and compile code for the ESP8266, right?

Install the ESP8266 tool chain

A tool chain is the set of tools that lets you compile and create binaries for a certain platform. Since we want to create binaries for the ESP8266 we need a different tool chain than the one that comes with the plain vanilla Arduino IDE. To save you the hassle of downloading many different files and copying them into obscure locations, the Arduino IDE has a wonderful feature: the Board Manager. It lets you install support for many different chips and boards with just a few clicks. But first of all we have to tell the Arduino IDE where it should look for board definitions:

Open the Arduino IDE

- Go to your preferences/settings and in the text box `Additional Board Manager URLs` enter this URL: http://arduino.esp8266.com/stable/package_esp8266com_index.json
- Now go to `Tools > Board: … > Boards Manager…`, search for the ESP8266 board and click `Install`.
- Get a coffee and wait until it finishes.



Select the ESP8266 platform from the board manager

From time to time you want to come back to the Board Manager and make sure that you have the latest version of the ESP8266 tool chain installed. To do that simply click on the ESP8266 entry and select the latest version from the dropdown. Then click `Update`.

Selecting the Correct Board

Now your Arduino IDE knows about ESP8266 boards in general. But not all the ESP8266 boards are the same; there are subtle but important differences in available Flash Memory and how they can be programmed. The selection of the correct board also defines the names of the GPIO pins: the designers of the NodeMCU decided to introduce a completely new naming scheme for the pins. Instead of calling them `GPIO1`, `GPIO2` etc they decided to give them different numbers by using a "D"-prefix. So `D0` is `GPIO16`, `D1` is `GPIO5` and so on. By selecting a NodeMCU board you automatically have the D naming scheme available, and this helps a lot since these names are also printed on the module board.

So let's pick the correct board. If you bought the original Squix Starter Kit you will have to choose a `NodeMCU 1.0`: Go to `Tools > Board: * > NodeMCU 1.0 (ESP-12E Module)`

There is a plentitude of modules available. Please make sure that you have the correct board selected before you continue.

Setting the Correct Port

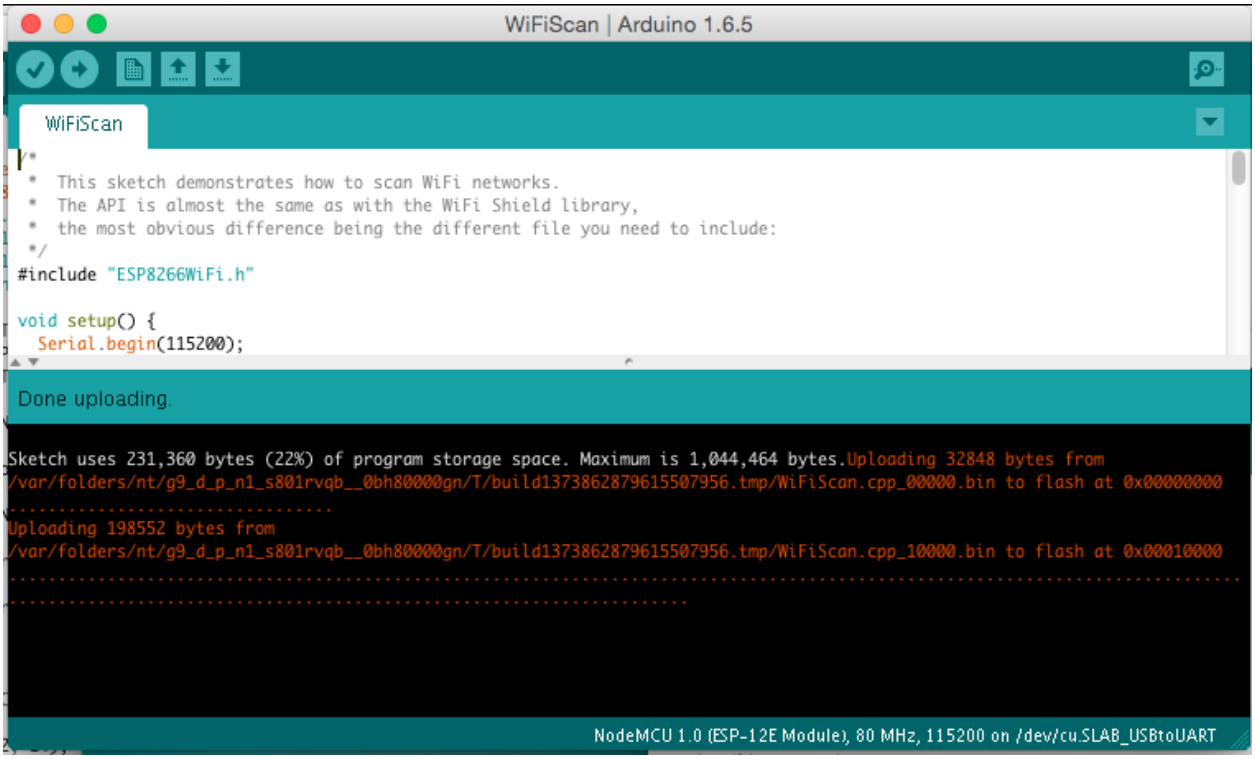
Serial interface: At the hardware level the ESP8266 is programmed through a serial interface. In short this is a very common communication interface which normally requires three lines: transmit (TX), receive (RX) and ground (GND). Both devices involved in the communication need to agree on the rate the characters are sent over the wire. This rate is usually measured in BAUD. 10 BAUD is equal to 1 character per second. Your average PC or Mac doesn't have such a serial interface, so how can we program the ESP8266? This is done through a Serial-to-USB converter. Some ESPs already come with a built-in converter; others need an external one for programming.

In an earlier step you already installed the drivers for this converter. If everything went well and the board is plugged into your computer you should now be able to select the serial connection. It should show up in the Menu under `Tools > Port`. On my Mac the device is called `/dev/cu.SLAB_USBtoUART`. On a PC it should be listed as a COM port labelled `COM#` (where `#` is some number).

If you cannot see a device that looks like the NodeMCU, try to unplug the ESP module and re-plug it after a few seconds. Also try a different USB socket. If that doesn't help consider restarting your computer… Make sure that you installed the driver as mentioned in the chapter about drivers.

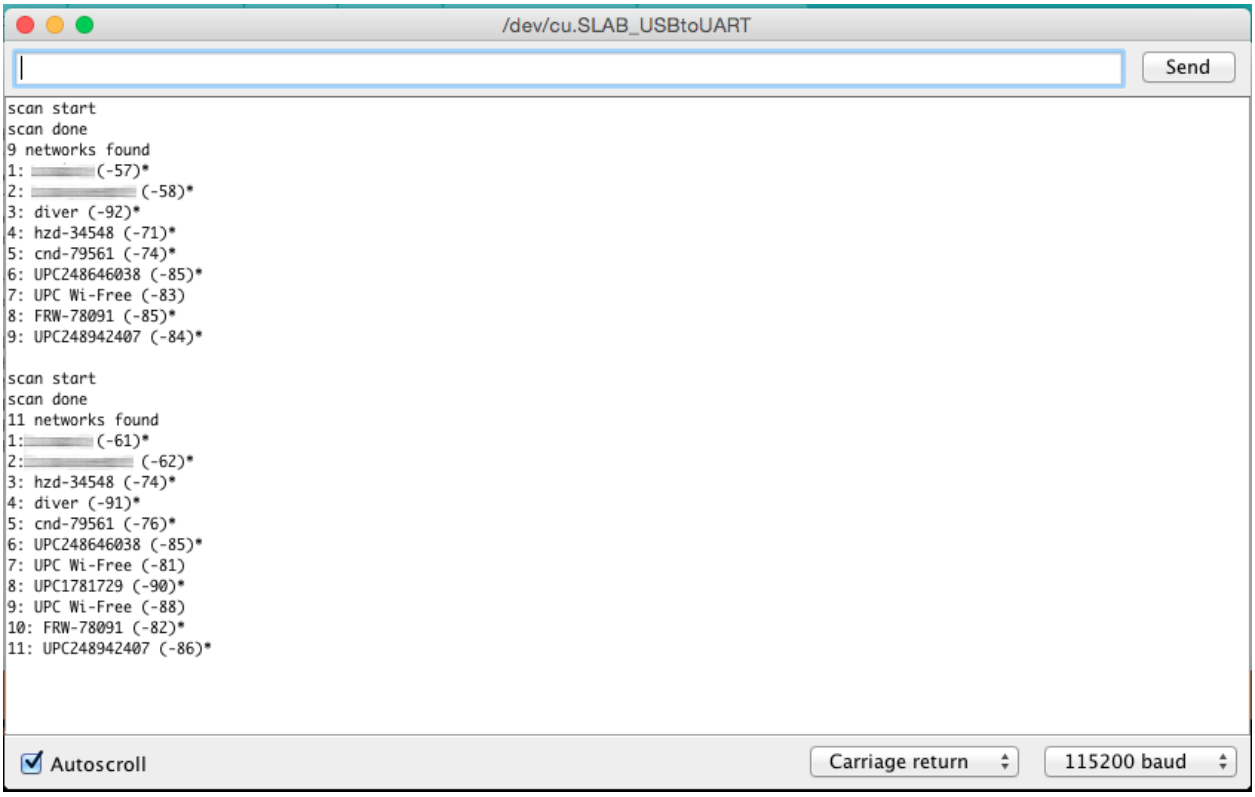
Testing the Setup: WiFi Scanner

Thanks for bearing with me until we get to the really cool part. We are going to run our first program on the NodeMCU! In the Menu of the Arduino IDE go to `File > Examples > ESP8266Wifi` and select `WiFiScan`. A new window will open up. This window is your current project and is also called a "Sketch". To compile and transfer the binary to the ESP8266 click on the green circle that contains an arrow on the very top of the window. If everything went well this will compile the sketch and upload the binary to the ESP. It might look something like this:



Wifi Scanner Output

If you see `Done uploading.` in the window, then click on the magnifying glass on the top right of the window. This is the serial console that you can use to see output from the NodeMCU module, or to send commands to the device. Make sure that the baud rate is set to `115200`. This rate is also set in the example code, and if you have a different setting the ESP will talk with a different speed than your PC listens. You can set the baud rate on the bottom left of the serial monitor. My output looks like this:



Serial Console of the Wifi Scanner

If you see something similar: congratulations! You have just set all the preconditions to run the WeatherStation code.

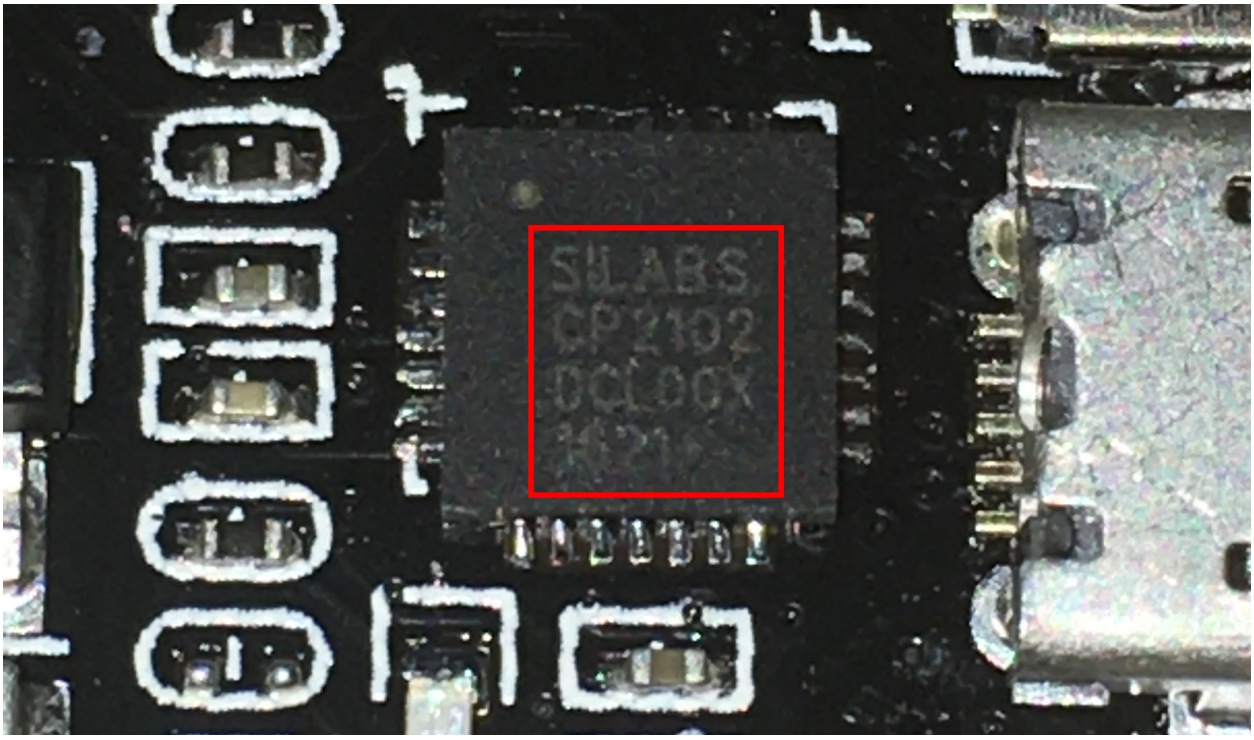
Trouble Shooting

Let me be honest: there are many reasons why this setup might not work. But don't give up so quickly! With a careful and analytical approach we will manage to get the ESP8266 running! The following paragraphs are structured by symptom and I will give you some ideas how to find the problem and how to solve it.

No serial port shows up after you connect the ESP8266 to your computer

This is a tough one because this is a symptom for many different causes:

- First please make sure that you have installed the correct driver, either for the CP2102 or the CH340. If you are not sure which one your ESP8266 has then better install both. The extra driver will only be used if you attach a matching hardware. The photo below shows how the CP2102 from Silabs looks like.



NodeMCU with a Silabs CP2102

- Another possible and frequent culprit is the USB cable. If you are sure that you installed the right drivers then try to use a different USB cable with the ESP8266. As a cross check you can also use the USB cable with another device (e.g. smartphone) and connect it to your PC. If the device is not recognized by your computer (and it is one that should be recognized) then throw the faulty cable away
- Sometimes it helps to restart your computer or choose another USB port. It happened to me several times that one USB port stopped working and only after a restart or changing the port the device would show up.
- It also happens relatively often that the NodeMCU is dead. But it is relatively hard to be 100% sure that it is really not working. If you previously didn't identify driver or cable as the cause for the problem we should focus on the NodeMCU module. Let's have a close look at the device. There are two LEDs: one on the ESP8266 module close to the antenna and the other one closer to the buttons. Do you see anything blink when you plug in the USB cable and connect it to your PC? If it blinks then the ESP8266 could be OK but the Serial-to-USB converter could be damaged. If there is no light then there are still many possibilities.

Failure during upload like `espcomm_upload_mem failed`

When you try to upload you see something like this in the console:

```
  warning: espcomm_sync failed
  error: espcomm_open failed
  error: espcomm_upload_mem failed
  error: espcomm_upload_mem failed
```

This means that for a number of reasons your computer could not upload the firmware to the NodeMCU. To understand what might be the cause we need to see what is happening during the upload of a new binary. Before the availability of easy-to-use developer modules like the NodeMCU you had to manually connect some pins of the ESP8266 to boot it into flash mode after a reset. This was very annoying since for every change in the code you had to compile, connect the pins, reset the ESP, wait until upload was complete, disconnect the boot mode pins and do a reset. Modules like the NodeMCU make this a lot easier since they have a special circuit which does all that when the serial-to-usb converter detects a special signal from your computer. Wonderful, right? Except: it doesn't always work. First let's try if the serial connection is working at all. Connect the NodeMCU to your computer and open the serial console. Now press the `RST` button and check what will be printed in the console. Depending on the selected transfer speed (lower right corner of the serial monitor) you either see strange characters or something similar to this:

```
ets Jan  8 2013,rst cause:2, boot mode:(3,6)

load 0x4010f000, len 1384, room 16
tail 8
chksum 0x2d
csum 0x2d
v3ffe85e8
~ld
```

I had to set the speed to 74880 baud to get this output. If you see this text then your computer and your ESP8266 can communicate with each other. Now we try to fix it by one of these measures:

- Press and hold the button labelled `FLASH` while pressing the button labelled `RST`. Then try again if the upload works. This button combination will manually set the ESP8266 into flash mode
- The settings in the Arduino Tool menu are also a frequent source of problems: have you selected the right board (e.g. NodeMCU V1.0) and the right USB/Serial port? Try also different upload speeds. The NodeMCU should automatically detect the requested transfer speed but this does not always work.

Summary

Before we continue to the WeatherStation project let's have a closer look at what we just accomplished:

1. We installed a driver which lets us program the ESP8266 with custom code that we wrote. Which driver needs to be installed depends on the Serial-to-USB converter we use. Some ESP modules already have such a converter; others will need an additional one.
2. We downloaded and installed the Arduino IDE. In the IDE we write the code, compile it and transfer it to the embedded device. If our code supports it we can even use the Serial Monitor to communicate with the device.
3. We used an example project, called a Sketch, to test our setup. The sample project installs firmware which uses the WiFi module to scan for available WiFi access points. It repeatedly writes this data to the serial line, and we can display it by opening the Serial Monitor tool. Remember, in a serial communication both parties need to agree on the speed the characters are getting transmitted. The example sets this to 115200 baud.