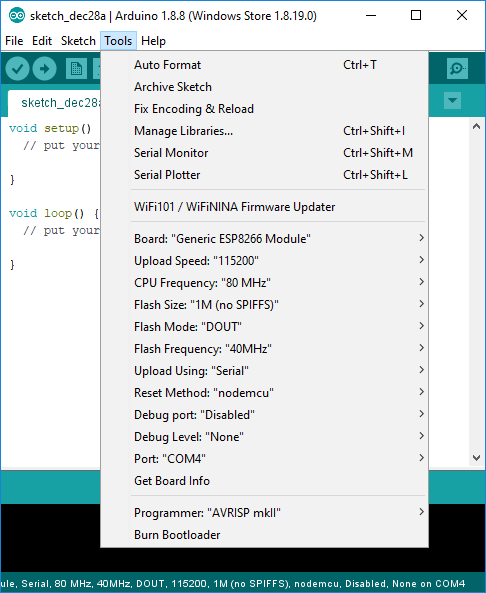
Build notes:

* Some extra solder paste on the ESP-07 pins might be useful. Try using a toothpick to manually add a small amount to each pin, or at least to the pins in use.
* Markings on the diodes correspond to the cathode, which is the straight line on the diode schematic symbol. So the line on the diode corresponds to the line on the diode schematic.
* Some extra solder might be useful on the two diodes – seems like the pads are too far separated. [2019-01-02] Found that there are two different footprints of 1N4148 diode. One is SOD-123 (correct), and the other is SOD-323 (too small for pad spacing).
* When attempting to solder the back side of the pin headers, try pushing a jumper onto the pin header to keep it snug to the board when inverting it to solder.

Test / troubleshooting:

* I needed to have a wifi antenna attached to be able to connect. Haven’t thoroughly understood this yet. [2018-12-31] Issue occurs even when not sitting on a conductive static bag. May be unique to module.
* Don’t expect “AT” commands to work with the original firmware. It’s sufficient if you are getting serial data out and the keyboard inputs can be seen echoed. If you see that, then both TX and RX serial are working.
* Use these settings rather than the ones on the arendts github site – they’re for Sonoff, not necessarily an EPS-07.  
  
* Make sure the reset method is “nodemcu”
* WiFi might not connect immediately after downloading. May need to unplug.

Tool Chain Notes

* Arduino 1.8.8 board packages are in Documents/ArduinoData
* OTA upload directory is subdirectory of the api directory: /var/www/html/api/Arduino
* Permissions need to have group write by www-data
* OTA server is uranus.davis.local. Need full DNS name.
* Need python installed at Windows level, not just in Ubuntu subsystem
* Ended up with Python 2.7.15, due to incompatibilities of pyCurl with 3.7.0. Pycurl must also be installed, pip install pycurl if the MSVCRT version of python is compatible with a wheel version of pycurl.
* Board package 2.4.2 seems to have an issue with DNS GetHostByName function. DNS works OK with 2.3.0, so reverted to that board package. [2018-12-31] DNS issue is that when decoding “.local” addresses, an mDNS message is sent rather than an ordinary DNS message. And there doesn’t appear to be any fallback to trying traditional DNS if no response on mDNS. Workaround for now is to hard-code IP addresses for MQTT, syslog, and OTA servers.
* To set up OTA in Arduino
  + Copy boards.txt and packages.txt from correct board version folder (e.g. 2.4.2) into packages\esp8266\hardware\esp8266\2.4.2
  + Copy espupload.py to packages\esp8266\hardware\esp8266\2.3.0\tools.
  + Edit HOST\_ADDR in espupload.py to server address where api/arduino directory is located (uranus.davis.local)

# Verbose Output from Successful Upload

PS C:\Users\jeff.DAVIS\Documents\Projects\WiFiRelay\sw\BoardTest> C:\Users\jeff.DAVIS\Documents\ArduinoData\packages\esp8266\tools\esptool\0.4.9\esptool.exe -cp COM4 -cb 115200 -cd nodemcu -ca 0x0 -bz 1M -bm dout -cf .\BoardTest.ino.bin -vv

esptool v0.4.9 - (c) 2014 Ch. Klippel <ck@atelier-klippel.de>

setting port from COM1 to COM4

setting baudrate from 115200 to 115200

setting board to nodemcu

setting address from 0x00000000 to 0x00000000

setting flash size from 512K to 1M

setting flash mode from qio to dout

espcomm\_upload\_file

espcomm\_upload\_mem

setting serial port timeouts to 1000 ms

opening bootloader

resetting board

trying to connect

flush start

setting serial port timeouts to 1 ms

setting serial port timeouts to 1000 ms

flush complete

espcomm\_send\_command: sending command header

espcomm\_send\_command: sending command payload

read 0, requested 1

trying to connect

flush start

setting serial port timeouts to 1 ms

setting serial port timeouts to 1000 ms

flush complete

espcomm\_send\_command: sending command header

espcomm\_send\_command: sending command payload

espcomm\_send\_command: receiving 2 bytes of data

espcomm\_send\_command: receiving 2 bytes of data

espcomm\_send\_command: receiving 2 bytes of data

espcomm\_send\_command: receiving 2 bytes of data

espcomm\_send\_command: receiving 2 bytes of data

espcomm\_send\_command: receiving 2 bytes of data

espcomm\_send\_command: receiving 2 bytes of data

espcomm\_send\_command: receiving 2 bytes of data

Uploading 239328 bytes from .\BoardTest.ino.bin to flash at 0x00000000

erasing flash

size: 03a6e0 address: 000000

first\_sector\_index: 0

total\_sector\_count: 59

head\_sector\_count: 16

adjusted\_sector\_count: 43

erase\_size: 02b000

espcomm\_send\_command: sending command header

espcomm\_send\_command: sending command payload

setting serial port timeouts to 15000 ms

setting serial port timeouts to 1000 ms

espcomm\_send\_command: receiving 2 bytes of data

writing flash

................................................................................ [ 34% ]

................................................................................ [ 68% ]

.......................................................................... [ 100% ]

starting app without reboot

espcomm\_send\_command: sending command header

espcomm\_send\_command: sending command payload

espcomm\_send\_command: receiving 2 bytes of data

closing bootloader

flush start

setting serial port timeouts to 1 ms

setting serial port timeouts to 1000 ms

flush complete