Sonoff TH01, TH16 flash with NodeMCU and installation of lua scripts

Flash with nodeMCU and upload .lua scripts steps

- build nodeMCU at https://nodemcu-build.com/
- download esptool.py from https://github.com/espressif/esptool
- download luatool.py - from https://github.com/4refr0nt/luatool
- place nodeMCU build, esptool.py, luatool.py and .lua scripts that you want to upload to Sonoff in one folder
- in init.lua script change SSID, APPWD, IPADR, IPROUTER, IPNETMASK to make for Sonoff device connecting to your local network possible.
- Install picocom on rpi (sudo apt-get install picocom)
- connect Sonoff pins with UART pins (3.3 to 3.3, Grnd to Grnd, RX to Tx, TX to RX)
- plug UART into RPi USB
- run esp in flash mode - hold sonoff button and connect power 3.3v cable to UART (usb serial converter)
- On RPi: Open terminal -> **cd “folder where all files are placed”**
- On RPi: **python esptool.py -p /dev/ttyUSB0 write_flash 0x000000 nodemcu-master-7-modules-2017-03-08-23-07-00-integer.bin** - (if connecting via TX & RX pins directly from RPi pins it would be AMA0 instead of USB0, of course file name should be changed to actual name of nodeMCU build file name in your case)
- restart Sonoff by unplugging it from power and then plugging in again - this time run normally without holding the button
- On RPi: **python luatool.py --delay 0.6 -p /dev/ttyUSB0 --src init.lua --dest init.lua**
- On RPi: **python luatool.py --delay 0.6 -p /dev/ttyUSB0 --src handler.lua --dest handler.lua**
- On RPi: Open picocom (115200 value can differ, if it does not work google & try other values) **sudo picocom /dev/ttyUSB0 --b 115200 --omap crcrlf --imap crcrlf**
- **node.restart()** - if scripts were transferred correctly and there are no errors you should see logs showing obtained IP
- (if reflashing or error looping its good to make file.format() before repeating the process)

After deployment

Now you should be able toggle Sonoff device using its button and also get to Sonoff using its statis IP that you set in init.lua script.

Json output

If url contains **check=1** (example http://sonoff_ip_address/?check=1) webpage will return json indicating current state of Sonoff wifi smart switch. Example: enabled switch will return {"pin1":1} where 1 indicates that its current status is ON. This mode is used to read switch status from remote services/automation systems.

UI output

If url contains **check=2** (example http://sonoff_ip_address/?check=2) webpage will display ON/OFF buttons which user can use to toggle the switch. Button indicating current state of the device will be marked with green border. This mode is useful when user wants to manually switch status.

Controlling via GET pin url request attributes

Sonoff device will change its state depending on url **pin** attribute. **pin=ON1** enables the Sonoff switch (example: http://sonoff_ip_address/?pin=ON1). **pin=OFF1** disables the Sonoff switch (example: http://sonoff_ip_address/?pin=OFF1).

Naturally **pin** attribute can be combined with **check** attribute (example: http://sonoff_ip_address/?check=1&pin=ON1) to toggle the switch and read its current state at the same time.

Tutorial

All steps have been recorded and put into this video: https://youtu.be/AlX1ZiVodwY