Install Mongoose OS on raspberry pi(including CLI MOS tool)

1. In your directory of choice (I typically of choice choose home) make a folder called ‘go’
2. You are now going to follow the manual build steps on the mongoose os github page. Essentially you just need to install the correct list of drivers, then clone the github mos repo into the go folder you just made.

The drivers you need are:

* Git – for cloning
* Go 1.13 or later (I just use the most up to date)
* Python3
* libftdi and libtdi1
* libusb 1.0
* pkg-config
* GNU Make – you get this from installing build-essential

To install these, you can use the apt-get command in terminal:

sudo apt-get install build-essential git golang-go python3 libftdi-dev libusb-1.0-0-dev pkg-config

I also then manually install libftdi1-dev using: sudo apt-get install libftdi1-dev

1. Now cd into your go folder and clone the repo

* git clone <https://github.com/mongoose-os/mos>
* cd mos

1. once you’re in the mos folder you just need to make the build. This takes a while so give it time.
2. One the package is installed, in the command line make sure you are within the ../go/mos/ folder and run ./mos help to verify the build functions correctly

You should now have a functional version of the build installed.

If you don’t want to have to navigate to the mos folder every time add the go folder to GOPATH and gopath/bin to PATH. Use the export commands found [here](http://www.briandegger.co.uk/mos-tool/) There are tutorials for how to do this online as well. \*\*\* NOTE: These exact commands only work if you have the same folder structure, they may need to be adjusted for your folder hierarchy \*\*\*

For Communicating with 5g Yetis VIA the Faceplate

For flashing a new faceplate FW you need connected a usb to serial converter cable. You need all 6 pins connected. The faceplate has the silkscreen markings on the board, you match the cables to the silk screen(rx 🡪 rx, tx 🡪 tx, etc.). For windows you are done you can now use commands like call state etc, however for linux and on the pi you need to physically disconnect the dtr pin. This is because when linux opens a serial port it sends a combination of dtr and rts which just so happens to be the reset combination designed into our implementation of the esp32 chip. So every command resets the chip, thus you need to disconnect the dtr pin.

For use in python scripts

The easiest way I have found to implement the mos functions into a python script is to use the subprocess module to pass string arguments to the command line

Ex.:

Import subprocess

subprocess.run([‘./mos’, ‘call’, ‘state’]) \*you will need to specify the full path to ./mos if the script is not in the same folder and you have not added the correct folders to the PATH variable as explained above.