Debian_Bullseye_pITX-MX8M-PLUS

Build and Install Debian Bullseye for *pITX-MX8M-PLUS* (Dual and Quad Core)

This document provides instructions for advanced users how Embedian offers patches and builds Debian Bullseye for Embedian's *pITX-MX8M-PLUS* product platform and how to install the images to bring the evaluation board up and running.

Our aim is to fully support our hardware through device drivers. We also provide unit tests so that testing a board is easy and custom development can start precisely. The recommended host environment is Ubuntu 16.04.

Availability

pITX-MX8M-PLUS from Embedian

Basic Resources

- AArch64 Cross Compiler
    - Linaro: https://launchpad.net/linaro-toolchain-binaries
- Bootloader
    - Das U-Boot – the Universal Boot Loader http://www.denx.de/wiki/U-Boot
    - Source – http://git.denx.de/?p=u-boot.git;a=summary
- Linux Kernel
    - Linus's Mainline tree: http://git.kernel.org/?p=linux/kernel/git/torvalds/linux.git;a=summary
    - Freescale Linux source tree: git://git.freescale.com/imx/linux-imx.git
    - Freescale BSP meta layer: git://git.freescale.com/imx/meta-fsl-bsp-release
    - OpenEmbedded/Yocto BSP layer for Freescale's ARM platform git://git.yoctoproject.org/meta-fsl-arm

    - Embedian pITX-MX8M-PLUS kernel source tree for linux pitx_8mp_lf-5.10.y: git@git.embedian.com:developer/pitx-fsl-linux-kernel.git
- ARM based rootfs
    - Debian Squeeze: http://www.debian.org/

Generating SSH Keys

We recommend you use SSH keys to establish a secure connection between your computer and Embedian Gitlab server. To download u-boot and kernel source codes from Embedian server. You need to register from Embedian's Gitlab server and put your ssh public key there. The steps below will walk you through generating an SSH key and then adding the public key to our Gitlab account.

Step 1. Check for SSH keys

First, we need to check for existing ssh keys on your computer. Open up Git Bash and run:

```
$ cd ~/.ssh
$ ls
# Lists the files in your .ssh directory
```

Check the directory listing to see if you have a file named either `id_rsa.pub` or `id_dsa.pub`. If you don't have either of those files go to **step 2**. Otherwise, you already have an existing keypair, and you can skip to **step 3**.

Step 2. Generate a new SSH key

To generate a new SSH key, enter the code below. We want the default settings so when asked to enter a file in which to save the key, just press enter.

```
$ ssh-keygen -t rsa -C "your_email@example.com"
# Creates a new ssh key, using the provided email as a label
# Generating public/private rsa key pair.
# Enter file in which to save the key (/c/Users/you/.ssh/id_rsa): [Press enter]
$ ssh-add id_rsa
```

Now you need to enter a passphrase.

```
Enter passphrase (empty for no passphrase): [Type a passphrase]
Enter same passphrase again: [Type passphrase again]
```

Which should give you something like this:

```
Your identification has been saved in /c/Users/you/.ssh/id_rsa.
Your public key has been saved in /c/Users/you/.ssh/id_rsa.pub.
The key fingerprint is:
01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your_email@example.com
```

Step 3. Add your SSH key to Embedian Gitlab Server

Copy the key to your clipboard.

```
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAABDAQABAAABAQDQUEnh8uGpfxaZVU6+uE4bsDrs/tEE5/BPW7jMAxak
6qgOh6nUrQGBWS+VxMM2un3KzwvLRJSj8G4TnTK2CSmlBvR+X8ZeXNTyAdaDxULs/StVhH+QRtFEGy4o
iMIzvIlTyORY89jzhIsgZzwr01nqoSeWWASd+59JWtFjVy0nwVNVtbek7NfuIGGAPaijO5Wnshr2uChB
Pk8ScGjQ3z4VqNXP6CWhCXTqIk7EQl7yX2GKd6FgEFrzae+5Jf63Xm8g6abbE3ytCrMT/jYy5OOj2XSg
6jlxSFnKcONAcfMTWkTXeG/OgeGeG5kZdtqryRtOlGmOeuQe1dd3I+Zz3JyT your_email@example.c
om
```

Go to Embedian Git Server. At Profile Setting --> SSH Keys --> Add SSH Key

Paste your public key and press "Add Key" and your are done.

Create build environment

Install required packages

On Ubuntu machine:

```
$ sudo apt-get install binfmt-support qemu qemu-user-static debootstrap kpartx \

lvm2 dosfstools gpart binutils git lib32ncurses5-dev python-m2crypto gawk wget \
git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat libsdl1.2-dev \
autoconf libtool libglib2.0-dev libarchive-dev python-git xterm sed cvs subversion \
coreutils texi2html bc docbook-utils python-pysqlite2 help2man make gcc g++ \
desktop-file-utils libgl1-mesa-dev libglu1-mesa-dev mercurial automake groff curl \
lzop asciidoc u-boot-tools mtd-utils device-tree-compiler
```

Deploy Sources

Download archive containing the build script and support files for building Debian Bullseye

```
$ cd ~/

$ git clone git@git.embedian.com:developer/smarc_mx8_debian.git pitx_mx8mp_debian -b
debian_bullseye_pitxmx8mp

$ cd ~/pitx_mx8mp_debian

$ MACHINE=pitximx8mp4g ./pitx_make_debian.sh -c deploy
```

This environment prepared to build.

> ⚠ If the LPDDR4 is 6GB on your *pITX-MX8M-PLUS*, the MACHINE name will be `pitximx8mp6g`
>
> If the LPDDR4 is 2GB on your *pITX-MX8M-PLUS*, the MACHINE name will be `pitximx8mp2g`

Make Debian

Build all

The internet connection in your host PC has to be available.

```
$ cd ~/pitx_mx8mp_debian
$ sudo MACHINE=pitximx8mp4g ./pitx_make_debian.sh -c all |& tee build.log
```

⚠ If the LPDDR4 is 6GB on your *pITX-MX8M-PLUS*, the MACHINE name will be `pitximx8mp6g`

If the LPDDR4 is 2GB on your *pITX-MX8M-PLUS*, the MACHINE name will be `pitximx8mp2g`

Build by parts

Build bootloader

```
$ cd ~/pitx_mx8mp_debian
$ sudo MACHINE=pitximx8mp4g ./pitx_make_debian.sh -c bootloader
```

Build kernel, dtb files and kernel modules

```
$ cd ~/pitx_mx8mp_debian
$ sudo MACHINE=pitximx8mp4g ./pitx_make_debian.sh -c kernel
$ sudo MACHINE=pitximx8mp4g ./pitx_make_debian.sh -c modules
```

Build rootfs

```
$ cd ~/pitx_mx8mp_debian
$ sudo MACHINE=pitximx8mp4g ./pitx_make_debian.sh -c rootfs
```

Pack rootfs

```
$ cd ~/pitx_mx8mp_debian
$ sudo MACHINE=pitximx8mp4g ./pitx_make_debian.sh -c rtar
```

⚠ If the LPDDR4 is 6GB on your *pITX-MX8M-PLUS*, the MACHINE name will be `pitxims8mp6g`

If the LPDDR4 is 2GB on your *pITX-MX8M-PLUS*, the MACHINE name will be `pitximx8mp2g`

Setup microSD card Automatically

```
$ cd ~/pitx_mx8mp_debian
$ sudo MACHINE=pitximx8mp4g ./pitx_make_debian.sh -c sdcard -d /dev/sdX
```

where "/dev/sdX" is the microSD block device in your host system. Set the SW2 port 1-3 as (ON ON ON). Insert the microSD card and you will see *pITX-MX8M-PLUS* booting with Debian Bullseye.

Selecting display configuration is a matter of selecting an appropriate DTB file.

All available DTB files are listed in the table below.

DTB File Name	Description
imx8mp-pitx.dtb	Device tree blob for no display configuration.
imx8mp-pitx-hdmi.dtb	Device tree blob for HDMI display configuration.
imx8mp-pitx-lvds.dtb	Device tree blob for LVDS display configuration.
imx8mp-pitx-m7.dtb	Device tree blob for Cortex-M7 co-processor configuration.

1. The default display output is hdmi. If you would like to change to default display output to different interfaces, make changes the file pitx_make_debian.sh and find `readonly DISPLAY="-lvds"`
   No Display: readonly DISPLAY=""
   HDMI: *readonly DISPLAY="-hdmi"*
   LVDS : *`readonly DISPLAY="-lvds"`*
   M7: *`readonly DISPLAY="-m7"`*

Setup SD Card Manually

For these instruction, we are assuming: DISK=/dev/mmcblk0, "lsblk" is very useful for determining the device id.

```
$ export DISK=/dev/mmcblk0
```

Erase SD card:

```
$ sudo dd if=/dev/zero of=${DISK} bs=1M count=16
```

Create Partition Layout: Leave 2MB offset for flash.bin.

With util-linux v2.26, sfdisk was rewritten and is now based on libfdisk.

sfdisk
```
$ sudo sfdisk --version
```
sfdisk from util-linux 2.27.1

Create Partitions:

sfdisk >=2.26.x
```
$ sudo sfdisk ${DISK} <<-__EOF__
2M,48M,0x83,*
50M,,,
__EOF__
```

sfdisk <=2.25

```
$ sudo sfdisk --in-order --Linux --unit M ${DISK} <<-__EOF__
2,48,0x83,*
,,,-
__EOF__
```

Format Partitions:

```
for: DISK=/dev/mmcblk0
$ sudo mkfs.vfat -F 16 ${DISK}p1 -n boot
$ sudo mkfs.ext4 ${DISK}p2 -L rootfs

for: DISK=/dev/sdX
$ sudo mkfs.vfat -F 16 ${DISK}1 -n boot
$ sudo mkfs.ext4 ${DISK}2 -L rootfs
```

Mount Partitions:

On some systems, these partitions may be auto-mounted...

```
$ sudo mkdir -p /media/boot/
$ sudo mkdir -p /media/rootfs/

for: DISK=/dev/mmcblk0
$ sudo mount ${DISK}p1 /media/boot/
$ sudo mount ${DISK}p2 /media/rootfs/

for: DISK=/dev/sdX
$ sudo mount ${DISK}1 /media/boot/
$ sudo mount ${DISK}2 /media/rootfs/
```

Install Boot File

Users need to set SW2 port 1-3 as (ON ON ON). In this way, *pITX-MX8M-PLUS* will always boot up from microSD card.

Fuse flash.bin to the microSD card.

~/

```
$ cd ~/pitx_mx8mp_debian
$ sudo dd if=output/imx-boot-sd.bin of=${DISK} bs=1024 seek=32
```

uEnv.txt based bootscript

Copy uEnv.txt to the boot partition:

~/smarc_mx8_debian

```
$ sudo cp -v embedian/uEnv.txt /media/boot/
```

Install Kernel Image

Copy Image to the boot partition:

~/smarc_mx8_debian

```
$ sudo cp -v output/Image /media/boot
```

Install Kernel Device Tree Binary

```
$ sudo mkdir -p /media/boot/dtbs

$ sudo cp -v output/<device tree name> /media/boot/dtbs/imx8mp-pitx.dtb
```

Selecting display configuration is a matter of selecting an appropriate DTB file.

All available DTB files are listed in the table below.

DTB File Name	Description
imx8mp-pitx.dtb	Device tree blob for no display configuration.
imx8mp-pitx-hdmi.dtb	Device tree blob for HDMI display configuration.
imx8mp-pitx-lvds.dtb	Device tree blob for LVDS display configuration.
imx8mp-pitx-m7.dtb	Device tree blob for Cortex-M7 co-processor configuration.

The device tree name in your SD card has be to imx8mp-pitx.dtb

Install Root File System and Kernel Modules

Extract Root File System:

directory where your root file system is

```
$ sudo tar xvfz output/rootfs.tar.gx -C /media/rootfs
```

Note

1. MAC address is factory pre-installed at on board I2C EEPROM at offset 60 bytes). It starts with Embedian's vendor code *10:0D :32*. u-boot will read it and pass this parameter to kernel.
2. Kernel module is pre-built in debian rootfs

Remove SD card:

```
$ sync
$ sudo umount /media/boot
$ sudo umount /media/rootfs
```

Build Results

The resulted images are located in *~/pitx_mx8mp_debian/output* directory:

Image Name	Description
rootfs.tar.gz	Root filesystem tarball for installation on SD card and eMMC
Image	Linux Kernel Image
imx-boot-sd.bin	Boot file for SD card and eMMC

DTB File Name	Description
imx8mp-pitx.dtb	Device tree blob for no display configuration.
imx8mp-pitx-hdmi.dtb	Device tree blob for HDMI display configuration.
imx8mp-pitx-lvds.dtb	Device tree blob for LVDS display configuration.
imx8mp-pitx-m7.dtb	Device tree blob for Cortex-M7 co-processor configuration.

Linux Console Access

User Name	User Password	User Descriptor
root	root	system administrator
user	user	local user
x_user		used for X session access

Setup eMMC

Set SW2 port 1-3 as (ON ON ON) and boot up from your microSD card. Run the following command as ***root*** user.

```
$ install_debian.sh -d <-hdmi/-lvds>
```

1. The "-d" parameter specifies which device tree blob that you would like copy to eMMC.
   No Display: readonly DISPLAY=""
   LVDS: *-d -lvds*
   HDMI: *-d -hdmi*

Modify the kernel configuration

To modify the kernel configuration (add/remove features and drivers). Please follow the step below.

```
$ cd ~/pitx_mx8mp_debian/src/kernel
$ sudo make arch=arm64 mrproper
$ sudo make arch=arm64 pitximx8mp_defconfig
$ sudo make arch=arm64 menuconfig

Navigate the menu and select the desired kernel functionality.
Exit the menu and answer "Yes" when asked "Do you wish to save your new configuration?"
$ sudo make ARCH=arm64 savedefconfig
$ sudo cp arch/arm64/configs/pitximx8mp_defconfig arch/arm64/configs/pitximx8mp_defconfig.orig
$ sudo cp .config arch/arm64/configs/pitximx8mp_defconfig
Follow the instructions above to rebuild kernel and modules, repack rootfs images and recreate SD card
```

Video Decoding

For playing video, we can use three solutions to support it.

a) `# gplay-1.0 <video file>`

b) `# gst-launch-1.0 playbin uri=file://<video absolute path>`

c) (i) if video container on .mp4 format

```
    # gst-launch-1.0 filesrc location=<file name.mp4> typefind=true ! video/quicktime ! qtdemux ! queue
max-size-time=0 ! vpudec ! queue max-size-time=0 ! kmssink force-hantrope=true sync=false &
```

(ii) if video container on .ts format

```
    # gst-launch-1.0 filesrc location=<file name.ts> typefind=true ! video/mpegts ! tsdemux ! queue
max-size-time=0 ! vpudec ! queue max-size-time=0 ! waylandsink
```

WiFi

The Debian Bullseye release includes NXP 88W8997 wifi chipset. Users can choose mPCIe or M.2 key E form factor wifi modules based on NXP 88W8997 chipset.

M.2 Form Factor:

- AzureWave P/N: AW-CM276MA-PUR
- Laird Connectivity P/N: 60-2230C
- Embedded Artists 1YM M.2 Module

mPCIe Factor:

- Globascale Technologies NXP 88W8997 2x2 WiFi 802.11ac+BT 5.0 mini PCIe Card w/ Two External SMA Antennas

Boot up the device and load the driver modules in the kernel.

```
root@pitximx8mp4g:~# modprobe moal mod_para=nxp/wifi_mod_para.conf
[ 33.834782] can2-stby: disabling
[ 33.838051] VSD1_3V3: disabling

[ 33.979809] wlan: Loading MWLAN driver
[ 33.984701] wlan_pcie 0000:01:00.0: enabling device (0000 -> 0002)
[ 33.991014] Attach moal handle ops, card interface type: 0x204
[ 34.000829] PCIE8997: init module param from usr cfg
[ 34.005845] card_type: PCIE8997, config block: 0
[ 34.010483] cfg80211_wext=0xf
[ 34.013465] wfd_name=p2p
[ 34.016011] max_vir_bss=1
[ 34.018632] cal_data_cfg=none
[ 34.021611] drv_mode = 7
[ 34.024159] ps_mode = 2
[ 34.026604] auto_ds = 2
[ 34.029084] fw_name=nxp/pcieuart8997_combo_v4.bin
[ 34.033830] rx_work=1 cpu_num=4
[ 34.037010] Attach mlan adapter operations.card_type is 0x204.
[ 34.046917] Request firmware: nxp/pcieuart8997_combo_v4.bin
[ 35.013725] FW download over, size 627620 bytes
[ 35.879247] WLAN FW is active
[ 35.882226] on_time is 35807347500
[ 35.917890] fw_cap_info=0x18fcffa3, dev_cap_mask=0xffffffff
[ 35.923500] max_p2p_conn = 8, max_sta_conn = 8
[ 35.956580] wlan: version = PCIE8997-16.68.10.p16-MXM5X16214-GPL-(FP92)
[ 35.966307] wlan: Driver loaded successfully
root@pitximx8mp4g:~#
```

Verify that the module is now visible to the system.

```
root@pitximx8mp4g:~# ifconfig -a
can0: flags=128<NOARP> mtu 16
unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 10 (UNSPEC)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 35

can1: flags=128<NOARP> mtu 16
unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 10 (UNSPEC)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 36

eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
ether 10:0d:32:01:00:01 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
ether 10:0d:32:02:00:01 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 54

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 3452 bytes 216146 (211.0 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 3452 bytes 216146 (211.0 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
ether 4a:6b:15:b3:7f:a4 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

p2p0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
ether 2a:08:86:b1:27:cb txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

uap0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
ether 5a:57:c4:46:2b:68 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@pitximx8mp4g:~#
```

Get the related packages

```
root@pitximx8mp4g:~# sudo apt-get update
```

```
root@pitximx8mp4g:~# sudo apt-get install wireless-tools udhcpc vim
```

In case you need to see which network and you can scan it and select the one you need.

```
root@pitximx8mp4g:~# iwlist mlan0 scan
mlan0 Scan completed :
Cell 01 - Address: D8:FE:E3:5F:68:98
ESSID:"Risetek"
Mode:Master
Frequency=2.412 GHz (Channel 1)
```

Identify the network and add it to the WPA supplicant file.

```
root@pitximx8mp4g:~# vim /etc/wpa_supplicant.conf
```

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
scan_ssid=1
ssid="embedian"
psk="xxxxxxxxxx"
}
```

Remove the wpa_supplicant that might already be in use.

```
root@pitximx8mp4g:~# rm /var/run/wpa_supplicant/mlan0
```

Associate the Wi-Fi with config

```
root@pitximx8mp4g:~# wpa_supplicant -B -i mlan0 -c /etc/wpa_supplicant.conf
Successfully initialized wpa_supplicant
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
nl80211: kernel reports: Match already configured
rfkill: Cannot open RFKILL control device
root@pitximx8mp4g:~#
```

Check if you have right SSID associated.

```
root@pitximx8mp4g:~# iwconfig mlan0
mlan0 IEEE 802.11-DS ESSID:"embedian" [14]
Mode:Managed Frequency=5.745 GHz Access Point: 48:EE:0C:ED:D7:38
Bit Rate:6.5 Mb/s Tx-Power=24 dBm
Retry limit:9 RTS thr=2347 B Fragment thr=2346 B
Encryption
key:****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-
****-****-****-****-****-****-****-****-****-****-**** Security mode:open
Power Management:off
Link Quality=3/5 Signal level=-66 dBm Noise level=-91 dBm
Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:27439
Tx excessive retries:8 Invalid misc:24 Missed beacon:0

root@pitximx8mp4g:~#
```

Use DHCP to get IP

```
root@pitximx8mp4g:~# udhcpc -i mlan0
udhcpc: started, v1.32.0
udhcpc: sending discover
udhcpc: sending select for 192.168.1.57
udhcpc: lease of 192.168.1.57 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 192.168.1.254
root@pitximx8mp4g:~#
```

You should be able to ping local network now.

```
root@pitximx8mp4g:~# ping 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=64 time=2141 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=64 time=1120 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=64 time=95.7 ms
64 bytes from 192.168.1.10: icmp_seq=4 ttl=64 time=1.63 ms
```

Modify /etc/resolv.conf of your preference, you will be able to ping out.

```
root@pitximx8mp4g:~# vim /etc/resolv.conf
```

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

```
root@pitximx8mp4g:/etc# ping www.google.com
PING www.google.com (172.217.163.36) 56(84) bytes of data.
64 bytes from maa05s01-in-f4.1e100.net (172.217.163.36): icmp_seq=1 ttl=117 time=7.23 ms
64 bytes from tsa01s13-in-f4.1e100.net (172.217.163.36): icmp_seq=2 ttl=117 time=39.7 ms
64 bytes from maa05s01-in-f4.1e100.net (172.217.163.36): icmp_seq=3 ttl=117 time=7.50 ms
64 bytes from tsa01s13-in-f4.1e100.net (172.217.163.36): icmp_seq=4 ttl=117 time=5.29 ms
64 bytes from tsa01s13-in-f4.1e100.net (172.217.163.36): icmp_seq=5 ttl=117 time=4.65 ms
64 bytes from tsa01s13-in-f4.1e100.net (172.217.163.36): icmp_seq=6 ttl=117 time=5.01 ms

--- www.google.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5010ms
rtt min/avg/max/mdev = 4.649/11.560/39.682/12.623 ms
```

version 1.0a, 08/18/2022

Last updated 2022-08-18