



EMBEDDED SYSTEMS I

INTRODUCTION TUTORIAL

STUDENT:
Pablo Vinícius da Silva Araújo

QUIXADÁ - CE

WARNING

This tutorial was made by just one computer engineering student, so it has never been verified by any professor or graduate computer engineer. Before you start, make sure that the tutorial may have flaws.

Welcome -

1 - Cross-Compiler installation

So before we start, I'll talk about some necessary information. Before you start configuring your environment, there are a few things you need to know. This tutorial will be more focused on Linux distributions and the board that we will be focused on is the BeagleBone Black. There are some examples on the internet with other operating systems and other cards, but they are few.

By the way, make sure your initial Linux configuration is right, when you are sure, start the following pass. The first thing we need to know is if your OS is a x86_64-linux or an aarch64-linux, you need to know about that before beginning your cross-compiler download.

Installing the cross-compiler. A cross-compiler is nothing more than a compiler for a different platform from the computer used for development. For example, a computer

with the x86 architecture using the cross-compiler ARM Cortex-A8 platform. In this tutorial we will be using the arm-none-eabi, which is the specific cross-compiler for hardware, and which has bare metal programming features.

Next stage, go to the webpage [Downloads | GNU Arm Embedded Toolchain Downloads – Arm Developer](#), there are some ARM GNU toolchains for you to make a download. In most cases, the user will make the download of the x86_64-linux so, let's continue.

First thing is access the Linux terminal and write a command to make a directory called 'lab' and, after this, enter it.

```
$ mkdir lab  
$ cd lab/
```

Inside this directory, make another directory, and this time, call him "toolchain", this directory will be the place where we will extract the cross-compiler download.

```
$ mkdir toolchain  
$ cd toolchain/  
$ tar jxvf gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2 -C  
toolchain/
```

Errors in this part : make sure about the cross-compiler version, the code line don't will be executed if the name of the archive has been changed.

Other one, check the path of your directories. After -C in the code line, make sure about the path, if you put it the wrong path, Linux will not find the archive.

You did it ?, let 's go.

Now we have to go open the “.bashrc”, this document is responsible for saving the Linux environment variables. Open a new terminal and put this.

```
$ gedit .bashrc
```

A window should appear for you, this window is the document .bashrc. Roll down the scroll mouse, in the end, put this line code and save the changes.

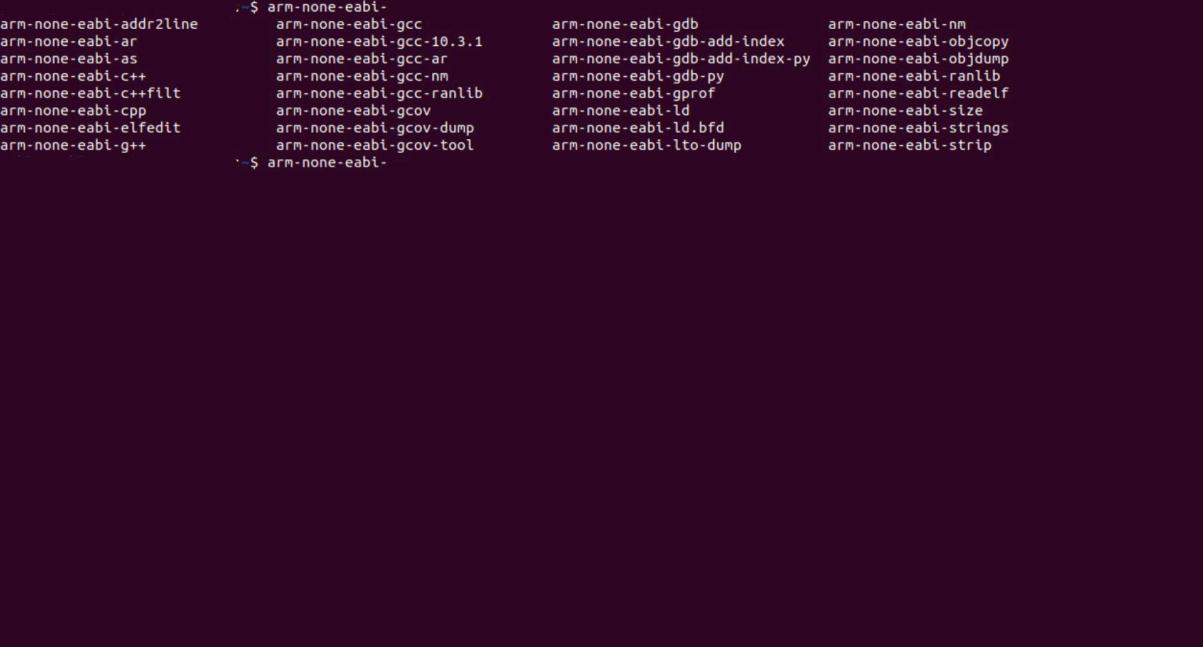
```
$ PATH=$PATH:~/lab/toolchain/gcc-arm-none-eabi-10.3-2021.10/bin
```

Right, so now, we are going to make sure about the cross compiler installation. Close your terminal and open him again. Put this code line and press TAB.

```
$ arm-none-eabi-
```

With this list of commands, it will be possible to check whether or not the cross compiler that was downloaded was in fact installed on your machine correctly.

If any type of error appears, try to redo the steps and interpret it according to the error that the Linux terminal shows you.

A terminal window showing a list of cross-compilers. The output is organized into three columns. The first column contains arm-none-eabi-addr2line, arm-none-eabi-ar, arm-none-eabi-as, arm-none-eabi-c++, arm-none-eabi-c++filt, arm-none-eabi-cpp, arm-none-eabi-elfedit, and arm-none-eabi-g++. The second column contains arm-none-eabi-gcc, arm-none-eabi-gcc-10.3.1, arm-none-eabi-gcc-ar, arm-none-eabi-gcc-nm, arm-none-eabi-gcc-ranlib, arm-none-eabi-gcov, arm-none-eabi-gcov-dump, and arm-none-eabi-gcov-tool. The third column contains arm-none-eabi-gdb, arm-none-eabi-gdb-add-index, arm-none-eabi-gdb-add-index-py, arm-none-eabi-gdb-py, arm-none-eabi-gprof, arm-none-eabi-ld, arm-none-eabi-ld.bfd, arm-none-eabi-lto-dump, arm-none-eabi-nm, arm-none-eabi-objcopy, arm-none-eabi-objdump, arm-none-eabi-ranlib, arm-none-eabi-readelf, arm-none-eabi-size, arm-none-eabi-strings, and arm-none-eabi-strip. A final prompt '\$ arm-none-eabi-' is at the bottom left.

```
$ arm-none-eabi-
arm-none-eabi-addr2line      arm-none-eabi-gcc          arm-none-eabi-gdb          arm-none-eabi-nm
arm-none-eabi-ar             arm-none-eabi-gcc-10.3.1   arm-none-eabi-gdb-add-index  arm-none-eabi-objcopy
arm-none-eabi-as             arm-none-eabi-gcc-ar       arm-none-eabi-gdb-add-index-py arm-none-eabi-objdump
arm-none-eabi-c++            arm-none-eabi-gcc-nm       arm-none-eabi-gdb-py        arm-none-eabi-ranlib
arm-none-eabi-c++filt        arm-none-eabi-gcc-ranlib   arm-none-eabi-gprof         arm-none-eabi-readelf
arm-none-eabi-cpp            arm-none-eabi-gcov        arm-none-eabi-ld           arm-none-eabi-size
arm-none-eabi-elfedit        arm-none-eabi-gcov-dump   arm-none-eabi-ld.bfd        arm-none-eabi-strings
arm-none-eabi-g++            arm-none-eabi-gcov-tool    arm-none-eabi-lto-dump     arm-none-eabi-strip
$ arm-none-eabi-
```

If your terminal starts to list a lot of cross-compilers, you did it, congratulations.

2 - Setting Environment TFTP

Trivial File Transfer Protocol (TFTP) provides a minimalist way to transfer files. It is generally used as a part of PXE boot or to update configuration or firmware on devices that have limited memory, such as routers, IP phones and embedded systems as a whole.

So, we are going to start setting up TFTP. First, put this code line in the terminal.

```
$ sudo apt-get install xinetd tftpd tftp
```

After that, enter in the directory /etc/xinetd.d. Next stage, we are going to create an archive called “tftp” and we are going to change permissions.

```
$ cd /etc/xinetd.d/  
$ sudo touch tftp  
$ ls -l
```

You will see the permissions in the whole archives.

```
$ sudo chmod 777 -R /etc/xinetd.d/tftp  
$ ls -l
```

```
:/etc/xinetd.d$ cd /etc/xinetd.d/  
:/etc/xinetd.d$ sudo touch tftp  
:/etc/xinetd.d$ ls -l  
total 48  
-rw-r--r-- 1 root root 640 fev 5 2018 chargen  
-rw-r--r-- 1 root root 313 fev 5 2018 chargen-udp  
-rw-r--r-- 1 root root 502 fev 5 2018 daytime  
-rw-r--r-- 1 root root 313 fev 5 2018 daytime-udp  
-rw-r--r-- 1 root root 391 fev 5 2018 discard  
-rw-r--r-- 1 root root 312 fev 5 2018 discard-udp  
-rw-r--r-- 1 root root 422 fev 5 2018 echo  
-rw-r--r-- 1 root root 304 fev 5 2018 echo-udp  
-rw-r--r-- 1 root root 312 fev 5 2018 servers  
-rw-r--r-- 1 root root 314 fev 5 2018 services  
-rwxrwxrwx 1 root root 0 abr 12 23:59 teste  
-rw-r--r-- 1 root root 0 abr 13 21:36 tftp ←————  
-rw-r--r-- 1 root root 569 fev 5 2018 time  
-rw-r--r-- 1 root root 313 fev 5 2018 time-udp  
:/etc/xinetd.d$ sudo chmod -R 777 /etc/xinetd.d  
:/etc/xinetd.d$ ls -l  
total 48  
-rwxrwxrwx 1 root root 640 fev 5 2018 chargen  
-rwxrwxrwx 1 root root 313 fev 5 2018 chargen-udp  
-rwxrwxrwx 1 root root 502 fev 5 2018 daytime  
-rwxrwxrwx 1 root root 313 fev 5 2018 daytime-udp  
-rwxrwxrwx 1 root root 391 fev 5 2018 discard  
-rwxrwxrwx 1 root root 312 fev 5 2018 discard-udp  
-rwxrwxrwx 1 root root 422 fev 5 2018 echo  
-rwxrwxrwx 1 root root 304 fev 5 2018 echo-udp  
-rwxrwxrwx 1 root root 312 fev 5 2018 servers  
-rwxrwxrwx 1 root root 314 fev 5 2018 services  
-rwxrwxrwx 1 root root 0 abr 12 23:59 teste  
-rwxrwxrwx 1 root root 0 abr 13 21:36 tftp ←————  
-rwxrwxrwx 1 root root 569 fev 5 2018 time  
-rwxrwxrwx 1 root root 313 fev 5 2018 time-udp  
:/etc/xinetd.d$
```

Check the permissions after and before the fourth command, if they change, you did it. Now, we are going to open this document with the following command.

```
$ gedit tftp
```

In the document, put this:

```
service tftp
{
protocol = udp
port = 69
socket_type = dgram
wait = yes
user = nobody
server = /usr/sbin/in.tftpd
server_args = /tftpboot
disable = no
}
```

Save your changes and close the terminal.

Now we are going to create the environment TFTPBoot.
Open the Linux terminal and put these code lines.

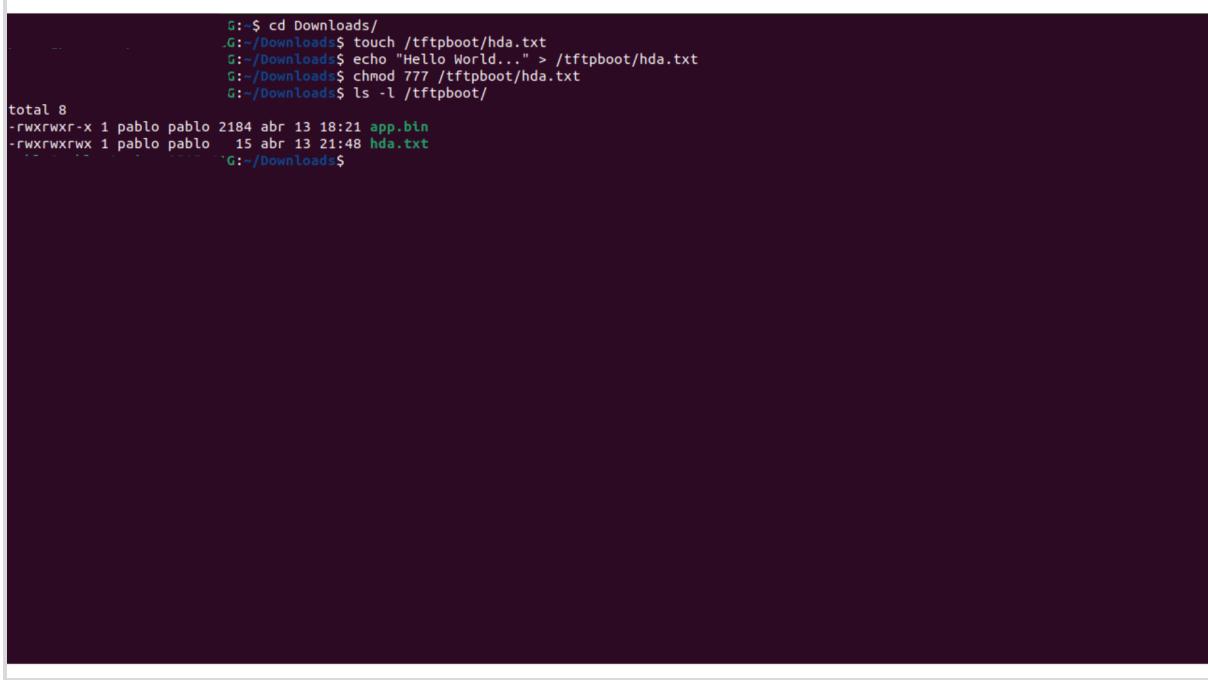
```
$ sudo mkdir /tftpboot
$ sudo chmod -R 777 /tftpboot
$ sudo chown -R nobody /tftpboot
```

Start the tftp through the xinetd.

```
$ sudo /etc/init.d/xinetd start
```

Now, test it. To test, let's do the following.

```
$ cd Downloads/  
$ touch /tftpboot/hda.txt  
$ echo "Hello World..." > /tftpboot/hda.txt  
$ chmod 777 /tftpboot/hda.txt  
$ ls -l /tftpboot/
```

A terminal window with a dark background and light-colored text. It shows a sequence of commands being run in a directory named 'Downloads'. The commands include navigating to the directory, creating a file named 'hda.txt', echoing text into it, changing its permissions to 777, and listing all files in the directory. The output shows two files: 'app.bin' and 'hda.txt'.

```
G: ~$ cd Downloads/  
.G: ~/Downloads$ touch /tftpboot/hda.txt  
.G: ~/Downloads$ echo "Hello World..." > /tftpboot/hda.txt  
.G: ~/Downloads$ chmod 777 /tftpboot/hda.txt  
.G: ~/Downloads$ ls -l /tftpboot/  
total 8  
-rwxrwxr-x 1 pablo pablo 2184 abr 13 18:21 app.bin  
-rwxrwxrwx 1 pablo pablo 15 abr 13 21:48 hda.txt  
G: ~/Downloads$
```

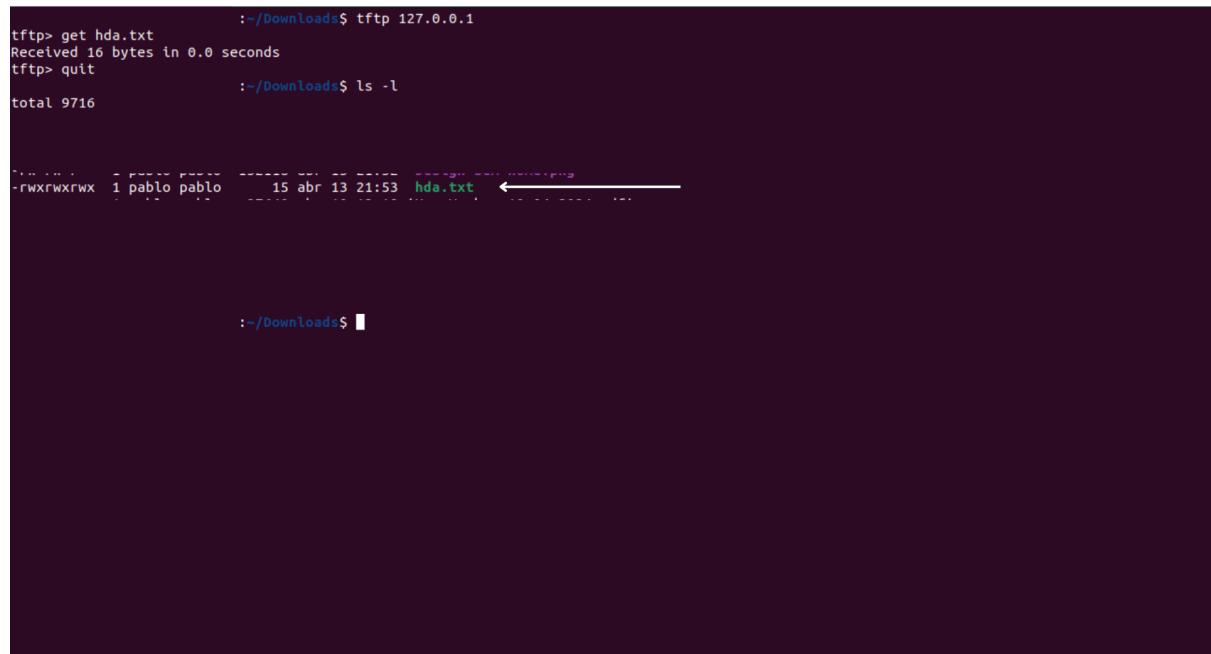
You will see the attributes, size, permissions etc...!

PS: try using sudo before setting the tftp address.

```
$ tftp 127.0.0.1 or $ sudo tftp 127.0.0.1  
$ tftp > get hda.txt  
Sent 722 bytes in 0.0 seconds - (aleatory byte size)  
$ tftp> quit  
$ ls -l
```

If you see the archive hda.txt in your Downloads folder, congratulations, your environment TFTP is working.

We can go to the next stage of the tutorial.



```
:/~/Downloads$ tftp 127.0.0.1
tftp> get hda.txt
Received 16 bytes in 0.0 seconds
tftp> quit
:/~/Downloads$ ls -l
total 9716
-rwxrwxrwx 1 pablo pablo 15 abr 13 21:53 hda.txt ←
```

The terminal session shows the user performing a TFTP download of a file named 'hda.txt' from the local host (127.0.0.1). After the download is complete, the user runs an 'ls -l' command to list the contents of the current directory, which includes the downloaded file 'hda.txt'. The file has permissions set to 'rwxrwxrwx' and was created by 'pablo' on April 13 at 21:53.

3 - Setting BeagleBone Black

Before starting, we must install the Minicom.

```
$ sudo apt install minicom
```

Now, take your BBB and connect the power cable to your computer, then take the TTL cable (Serial Cable), and plug the USB side in your computer and the other side, connect the pins.

Like this picture :



Right to left, put the black, jump two pins, put green and white.

Ok, now we are going to know what our serial cable is.

```
$ dmesg
```

Roll the scroll down, and take a printscreen. Remove your TTL and repeat the command above. Compare the prints and find your model cable. Most cases, it will be “pl 2303 ttyUSB0”.

See the pictures bellow:

```
[ 1872.629559] wlp2s0: authenticated
[ 1872.633040] wlp2s0: associate with 10:27:f5:07:c3:29 (try 1/3)
[ 1872.645332] wlp2s0: RX ReassocResp from 10:27:f5:07:c3:29 (capab=0x1431 status=0 aid=1)
[ 1872.647833] wlp2s0: associated
[ 1879.864113] wlp2s0: disconnect from AP 10:27:f5:07:c3:29 for new auth to 10:27:f5:07:c3:2a
[ 1879.930201] wlp2s0: authenticate with 10:27:f5:07:c3:2a
[ 1879.963006] wlp2s0: send auth to 10:27:f5:07:c3:2a (try 1/3)
[ 1879.964155] wlp2s0: authenticated
[ 1879.967661] wlp2s0: associate with 10:27:f5:07:c3:2a (try 1/3)
[ 1879.977328] wlp2s0: RX ReassocResp from 10:27:f5:07:c3:2a (capab=0x1411 status=0 aid=2)
[ 1879.979593] wlp2s0: associated
[ 1879.979775] ath: EEPROM regdomain: 0x804c
[ 1879.979780] ath: EEPROM indicates we should expect a country code
[ 1879.979782] ath: doing EEPROM country->regdmn map search
[ 1879.979783] ath: country maps to regdmn code: 0x3b
[ 1879.979784] ath: Country alpha2 being used: BR
[ 1879.979786] ath: Regpair used: 0x3b
[ 1879.979787] ath: regdomain 0x804c dynamically updated by country element
[ 1880.010981] wlp2s0: Limiting TX power to 30 (30 - 0) dBm as advertised by 10:27:f5:07:c3:2a
```

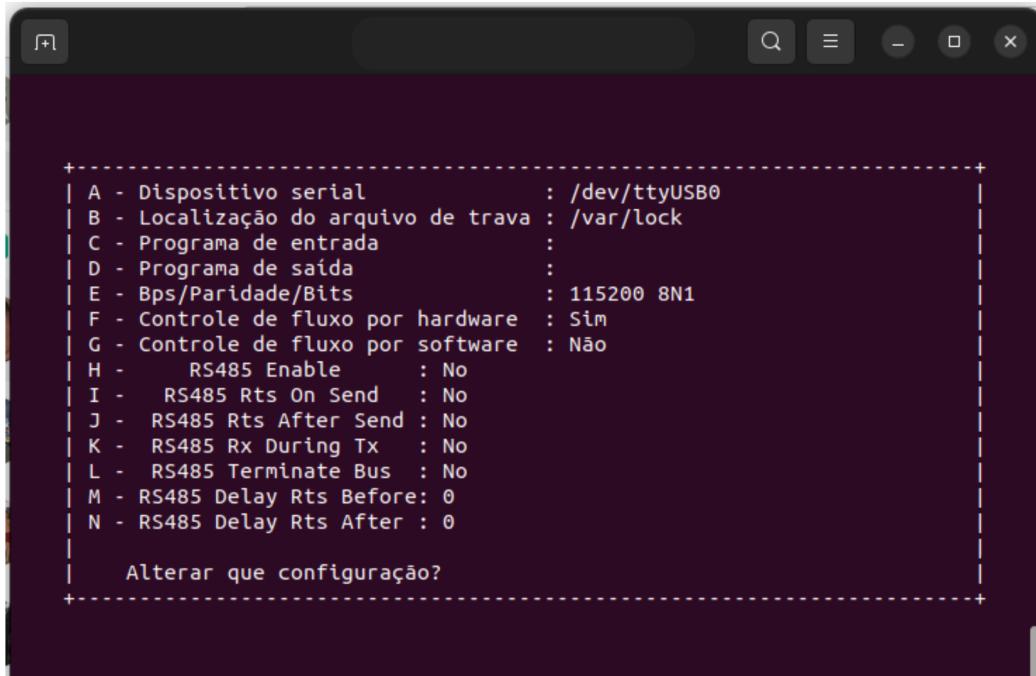
```
[ 1936.223244] wlp2s0: associated
[ 1936.223555] ath: EEPROM regdomain: 0x804c
[ 1936.223561] ath: EEPROM indicates we should expect a country code
[ 1936.223564] ath: doing EEPROM country->regdmn map search
[ 1936.223567] ath: country maps to regdmn code: 0x3b
[ 1936.223570] ath: Country alpha2 being used: BR
[ 1936.223573] ath: Regpair used: 0x3b
[ 1936.223576] ath: regdomain 0x804c dynamically updated by country element
[ 1936.329975] wlp2s0: Limiting TX power to 30 (30 - 0) dBm as advertised by 10:27:f5:07:c3:2a
[ 1937.951839] usb 1-1.2: USB disconnect, device number 7
[ 1937.952315] pl2303 ttyUSB0: pl2303 converter now disconnected from ttyUSB0
[ 1937.952363] pl2303 1-1.2:1.0: device disconnected
[ 1945.327363] wlp2s0: disconnect from AP 10:27:f5:07:c3:2a for new auth to 10:27:f5:07:c3:29
[ 1945.376214] wlp2s0: authenticate with 10:27:f5:07:c3:29
[ 1945.376338] wlp2s0: 80 MHz not supported, disabling VHT
[ 1945.408822] wlp2s0: send auth to 10:27:f5:07:c3:29 (try 1/3)
[ 1945.411044] wlp2s0: authenticated
[ 1945.414887] wlp2s0: associate with 10:27:f5:07:c3:29 (try 1/3)
[ 1945.427477] wlp2s0: RX ReassocResp from 10:27:f5:07:c3:29 (capab=0x1431 status=0 aid=1)
```

Now, open your minicom.

```
$ sudo minicom -s
```

In this window, find “Serial port configuration” and enter. Ok, press A and go to Serial Dispositiv and put /dev/"your TTL" ,

in my case I use `/dev/"ttyUSB0"`. Back, and save your changes in the minicom.



```
+-----+  
| A - Dispositivo serial      : /dev/ttyUSB0  
| B - Localização do arquivo de trava : /var/lock  
| C - Programa de entrada       :  
| D - Programa de saída        :  
| E - Bps/Paridade/Bits       : 115200 8N1  
| F - Controle de fluxo por hardware : Sim  
| G - Controle de fluxo por software  : Não  
| H -     RS485 Enable       : No  
| I -     RS485 Rts On Send   : No  
| J -     RS485 Rts After Send : No  
| K -     RS485 Rx During Tx  : No  
| L -     RS485 Terminate Bus : No  
| M - RS485 Delay Rts Before: 0  
| N - RS485 Delay Rts After : 0  
+-----+  
| Alterar que configuração?  
+-----+
```

4 - Setting the Server Communication

Before starting, we need to know what our current network interface is. Put this code line in the Linux terminal.

```
$ ifconfig
```

The terminal will show your interface network, take your netmask and save it.

After that, take an Ethernet cable and connect it to the BBB and to your computer. Go to the Ethernet Configuration and add an new ethernet network, put a name, go to IPV4, select Manual and in Address, put something like this:

Address	Netmask	Gateway
xx.x.x.x	your Netmask	xx.x.x.x

The Address and Gateway should be the same. Save them and close your configurations.

With your new connection created, select it in the network configuration.

Now, put this code line in the terminal Linux.

\$ sudo minicom

Restart your BBB and press space a few times, if you did, U-Boot will be shown to you. If you don't see U-Boot, repeat the process and restart again.

Congratulations, you stay in the U-boot of BeagleBone.

5 - Practicing

Make a download of this binary document:

□ [BinaryBBB](#)

Put it in your tftpboot folder. Open the minicom and enter in BeagleBone U-Boot.

```
BeagleBoard.org Debian Buster IoT Image 2020-04-06
Support: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian
default username:password is [debian:temppwd]

beaglebone login:
U-Boot SPL 2019.04-00002-g31a8ae0206 (May 13 2020 - 09:26:17 -0500)
Trying to boot from MMC2
Loading Environment from EXT4... Card did not respond to voltage select!

U-Boot 2019.04-00002-g31a8ae0206 (May 13 2020 - 09:26:17 -0500), Build: jenkins-github_Bootloader-Builder-139
CPU   : AM335X-GP rev 2.1
I2C:   ready
DRAM: 512 MiB
No match for driver 'omap_hsmmc'
No match for driver 'omap_hsmmc'
Some drivers were not found
Reset Source: Global external warm reset has occurred.
Reset Source: Power-on reset has occurred.
RTC 32KCLK Source: External.
MMC:  OMAP SD/MMC: 0, OMAP SD/MMC: 1
Loading Environment from EXT4... Card did not respond to voltage select!
Board: BeagleBone Black
<ethaddr> not set. Validating first E-fuse MAC
BeagleBone Black:
BeagleBone: cape eeprom: i2c_probe: 0x54:
BeagleBone: cape eeprom: i2c_probe: 0x55:
BeagleBone: cape eeprom: i2c_probe: 0x56:
BeagleBone: cape eeprom: i2c_probe: 0x57:
Net:   eth0: MII MODE
cpsw, usb_ether
Press SPACE to abort autoboot in 0 seconds
=>
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.8 | VT102 | Conectado 0:0 | ttyUSB0
```

=> **setenv serverip "your address"**

=> **setenv ipaddr "your address" - but, change the last number**

=> **ping "your address"**

If your host “Address” is alive, continue, if not check your connection with the Ethernet cable.

=> **tftp 0x80000000 /tftpboot/app.bin**

=> **go 0x80000000**

Your BeagleBone will start to show the binary numbers 0-15 in the board leds, Congratulations, you made this.

Possible errors: you entered an instruction name incorrectly, made a mistake when placing the addresses or placing the wrong binary path that you want to pass to the board.

```
CPU : AM335X-GP rev 2.1
I2C: ready
DRAM: 512 MiB
No match for driver 'omap_hsmmc'
No match for driver 'omap_hsmmc'
Some drivers were not found
Reset Source: Global external warm reset has occurred.
Reset Source: Power-on reset has occurred.
RTC 32KCLK Source: External.
MMC: OMAP SD/MMC: 0, OMAP SD/MMC: 1
Loading Environment from EXT4... Card did not respond to voltage select!
Board: BeagleBone Black
<ethaddr> not set. Validating first E-fuse MAC
BeagleBone Black:
BeagleBone: cape eeprom: i2c_probe: 0x54:
BeagleBone: cape eeprom: i2c_probe: 0x55:
BeagleBone: cape eeprom: i2c_probe: 0x56:
BeagleBone: cape eeprom: i2c_probe: 0x57:
Net: eth0: MII MODE
cpsw, usb_ether
Press SPACE to abort autoboot in 0 seconds
=> setenv serverip .
=> setenv ipaddr ...
=> tftp 0x80000000 /tftpboot/app.bin
link up on port 0, speed 100, full duplex
Using cpsw device
TFTP from server ; our IP address
Filename '/tftpboot/app.bin'.
Load address: 0x80000000
Loading: #
      355.5 KiB/s
done
Bytes transferred = 2184 (888 hex)
=> go 0x80000000
## Starting application at 0x80000000 ...
```

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.8 | VT102 | Conectado 0:4 | ttvUSB0

End.

For any questions, send an email to me:

pablovs28@outlook.com

Thanks for reading this, bye bye.

