**********Default**********

Re-look Paul's MQTT
Ultibo Australian user
ultibo-mqtt
11/07/22

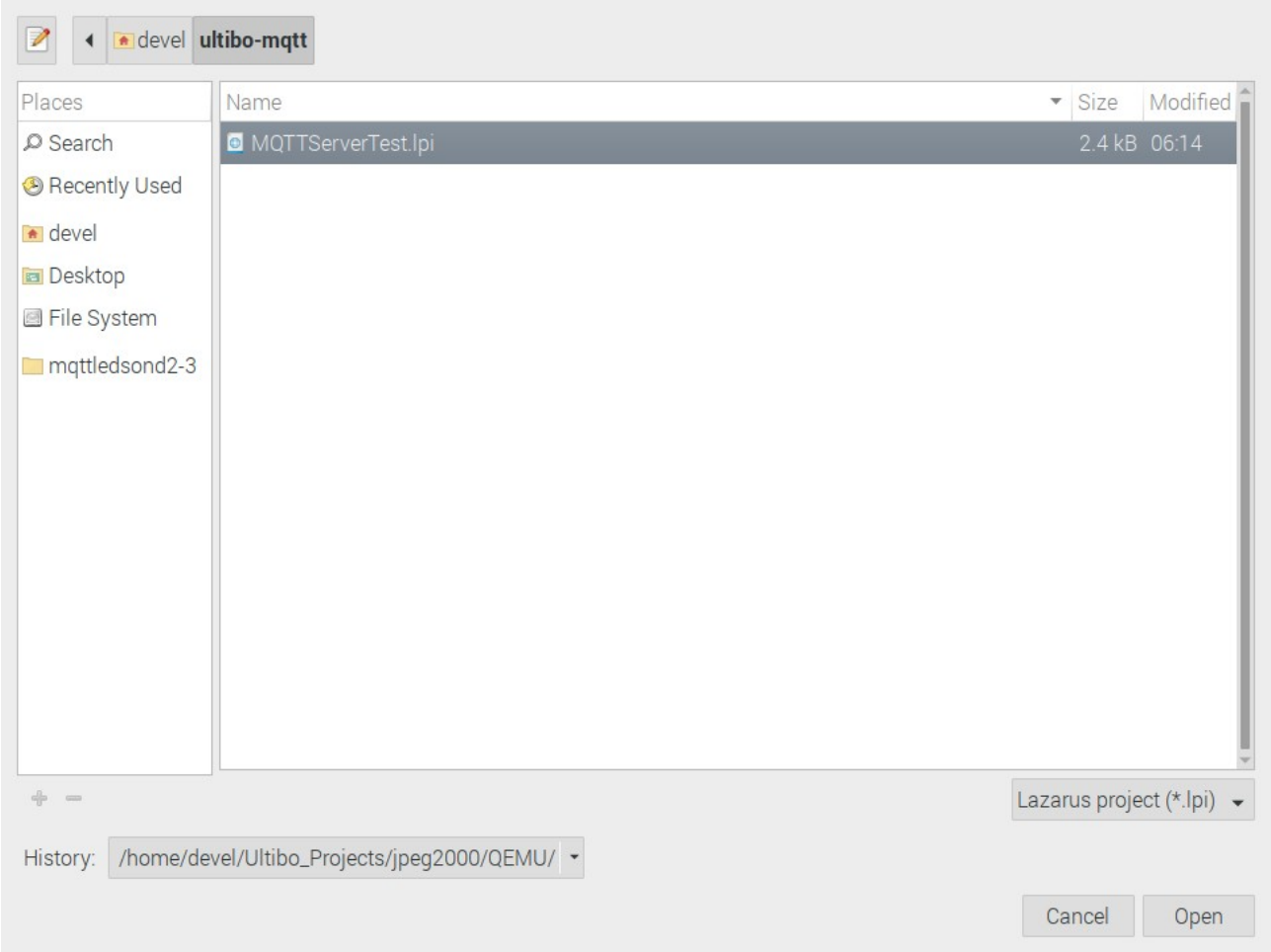
**********Default**********

This is a re-look at ultibo-mqtt, earlier work for use with Raspberry Pi Pico W project.

Goal: To see if the code as downloaded from https://github.com/pjde/ultibo-mqtt.git , from an Ultibo Australian user, compiles and runs on hardware. This will be used as a learning tool for understanding MQTT in Raspberry Pi Pico W project.
https://ultibo.org/forum/viewtopic.php?f=10&t=1427&p=9844&hilit=mqtt#p9844

"git clone https://github.com/develone/ultibo-mqtt.git"

With Lazarus IDE (Ultibo-Edition)



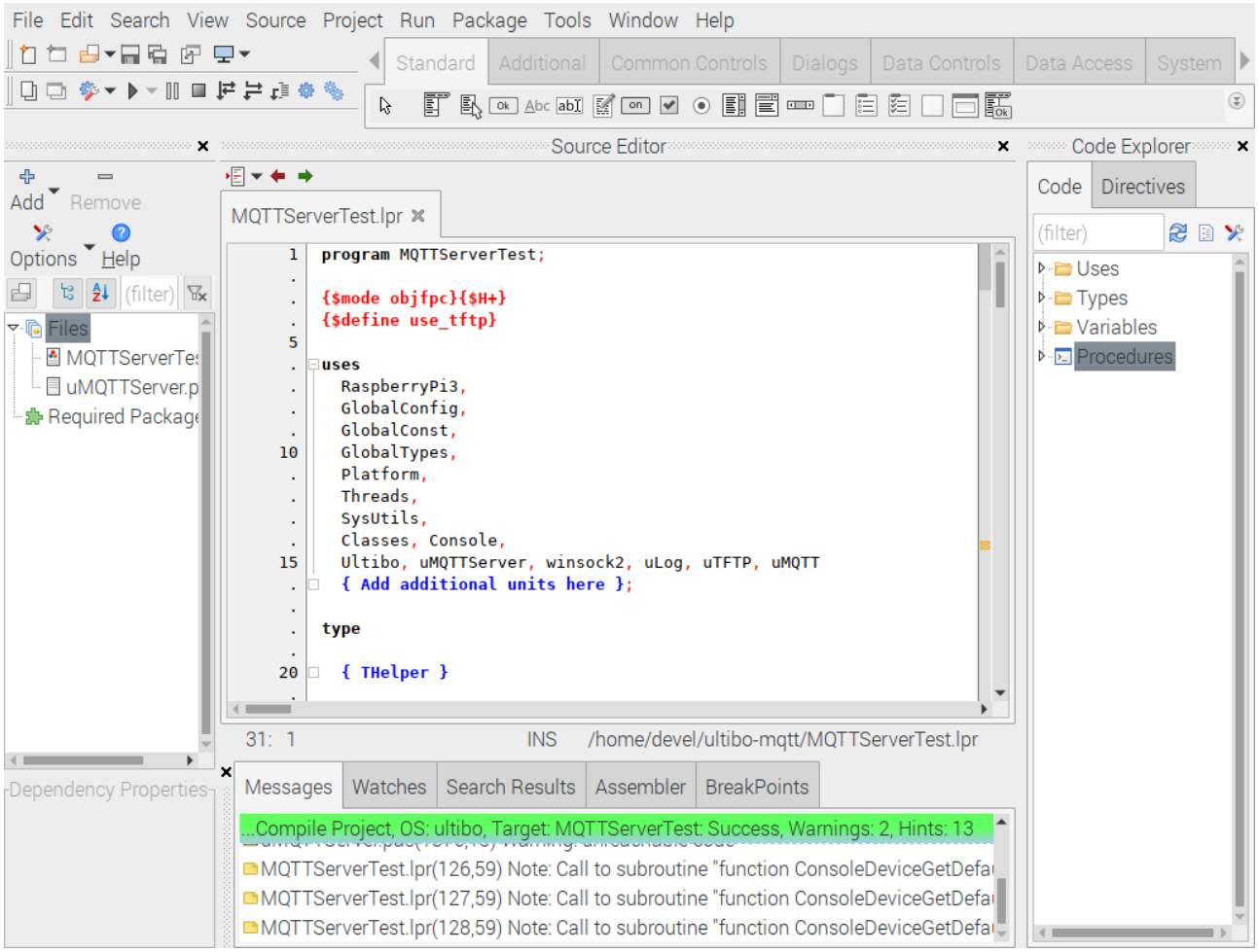
First step is to check that code compiles. From the main Tool Bar select Run/Compile. If the code is okay the green bar will appear. At first glance of the file **"MQTTServerTest.lpr"** after opening **"MQTTServerTest.lpi"** project file, is that code was intended for a **"RaspberryPi3"**. If the code is to be used on the **"RaspberryPi3" Bare Metal** you will need the firmware found at "git clone https://github.com/develone/firmwar_for_ultibo.git"

Ultibo Bare Metal only requires a few file.

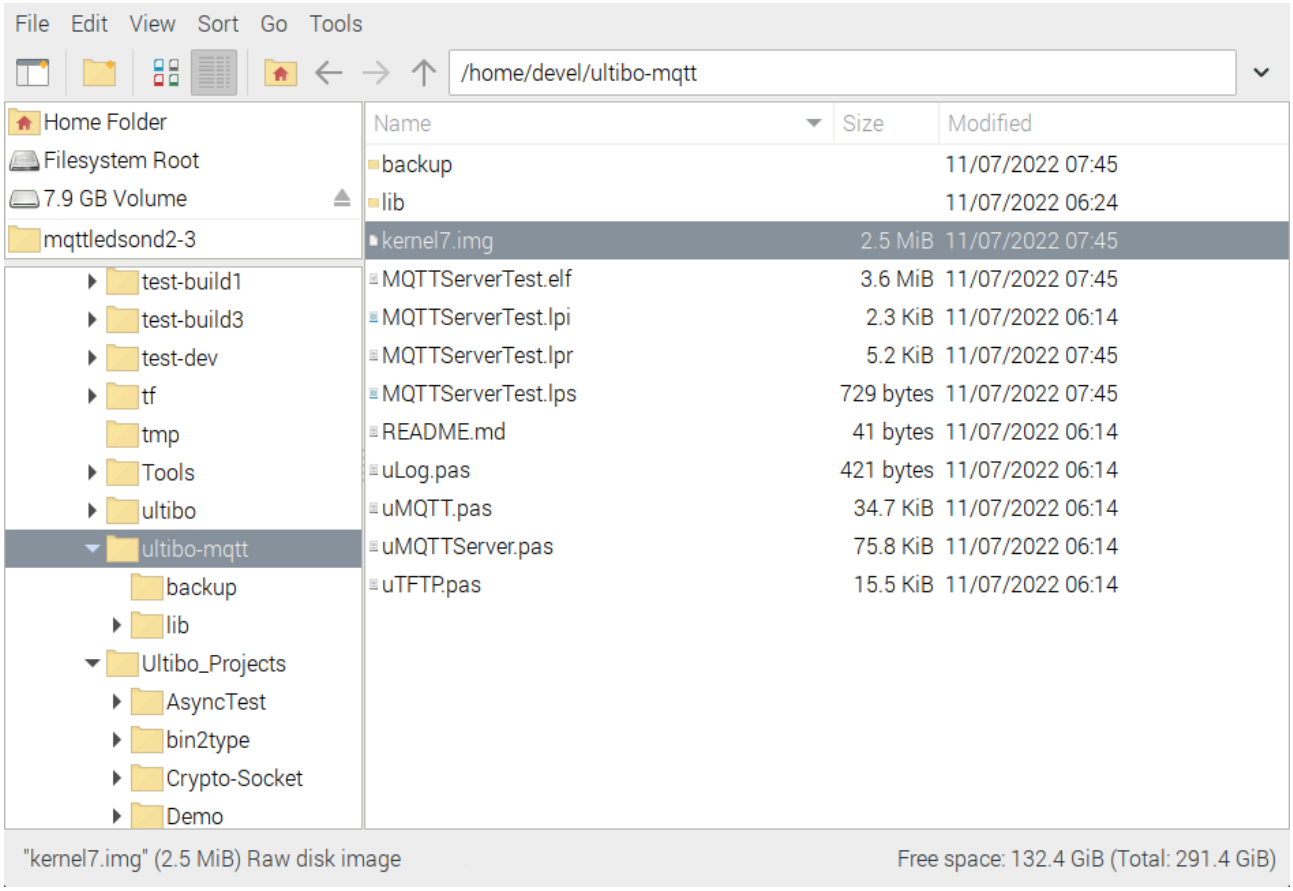
bootcode.bin
fixup.dat
fixup4.dat
start.elf
start4.elf
config.txt

The files above provide a way for the kernel to boot an have access to the hardware.

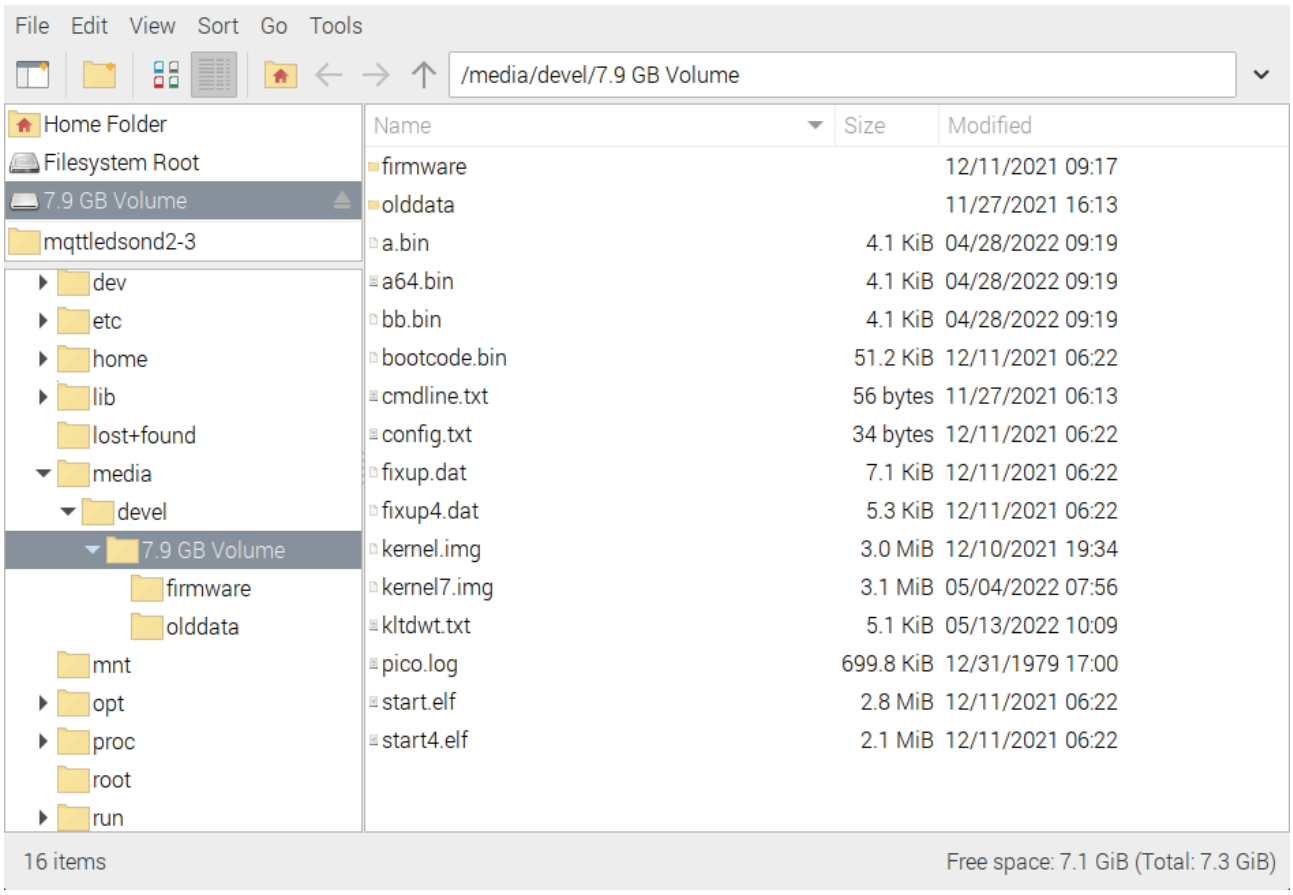
Most of code is contained in file kernel7.img.



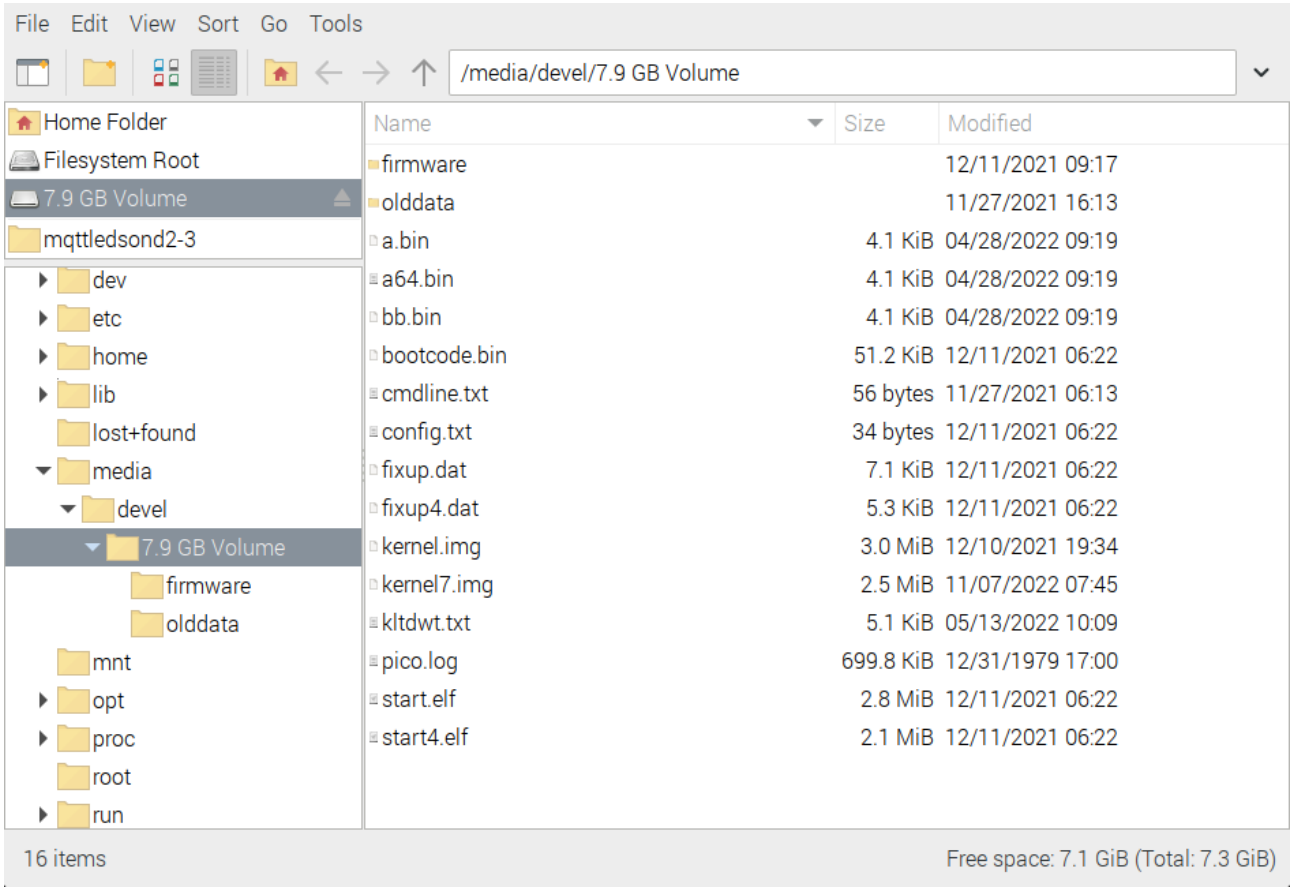
xx



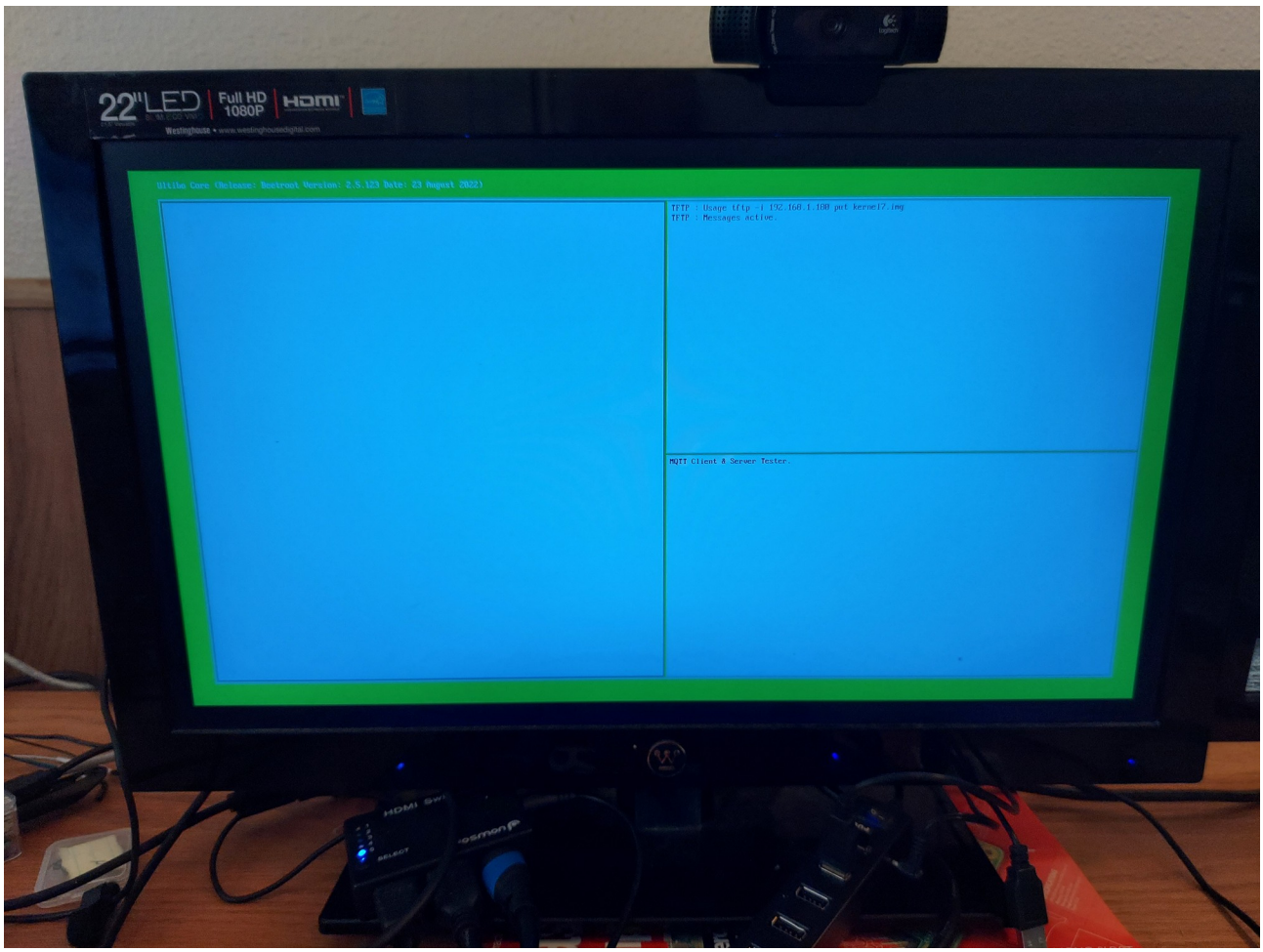
mount the micro sd



xx



Eject the micro sd and install on Raspberry Pi 2. Put the micro sd in the RPi2 and power up.

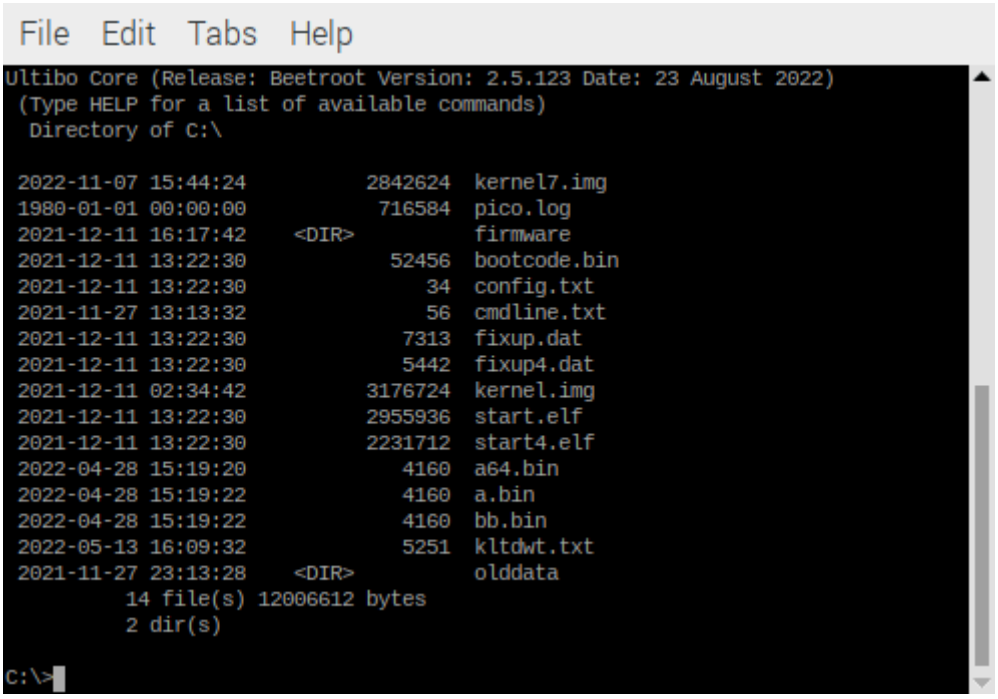


ping 192.168.1.180
PING 192.168.1.180 (192.168.1.180) 56(84) bytes of data.
64 bytes from 192.168.1.180: icmp_seq=2 ttl=128 time=1.88 ms
64 bytes from 192.168.1.180: icmp_seq=3 ttl=128 time=1.75 ms
c64 bytes from 192.168.1.180: icmp_seq=4 ttl=128 time=2.34 ms
64 bytes from 192.168.1.180: icmp_seq=5 ttl=128 time=1.88 ms
^C
--- 192.168.1.180 ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4052ms
rtt min/avg/max/mdev = 1.746/1.963/2.340/0.224 ms

Next we will add telnet to the project

{ needed for telnet }
Shell,
ShellFilesystem,
ShellUpdate,
RemoteShell,
{ needed for telnet }

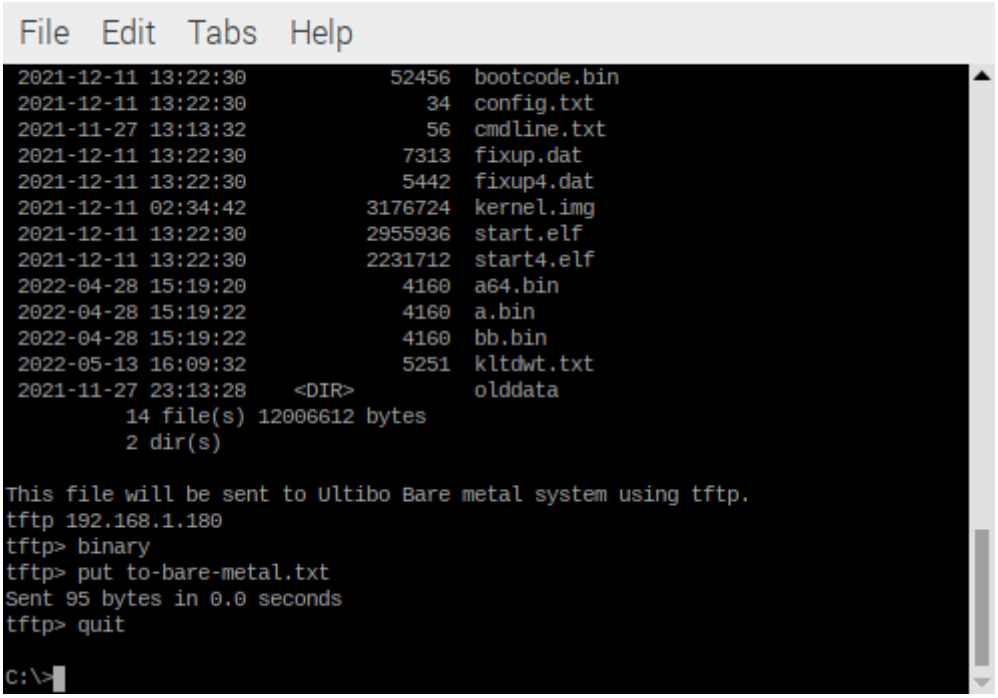
This now provides a telnet feature to our Bare Metal App.
“telnet 192.168.1.180”



To exit “logout”

Next we will test the tftp support which provides the transfer from the host to the Ultibo Bare Metal and from Ultibo Bare Metal to the host.
First we create an empty file with the command “touch to-bare-metal.txt”.
Add some text

tftp> put to-bare-metal.txt
Sent 95 bytes in 0.0 seconds
tftp> quit



xx

tftp 192.168.1.180
tftp> binary
tftp> put to-bare-metal.txt
Sent 163 bytes in 0.0 seconds

tftp> get config.txt
Received 34 bytes in 0.0 seconds
tftp> quit

cat config.txt
hdmi_force_hotplug=1
gpu_mem=128