

Creating Embedded Linux Solutions

Course Workbook

Table of Contents

About this Workbook	<u>Page 3</u>
Pre-Lab: Workshop Pre-requisites	<u>Page 4</u>
Lab 3: Creating Multi Processor Solutions	<u>Page 7</u>

About this Workbook

The contents of this workbook are created by Adiuvo Engineering & Training, Ltd.

If you have any questions about the contents, or need assistance, please contact Adam Taylor at adam@adiuvoengineering.com.

Pre-Lab

Creating Multi Processor Solutions

Required Hardware

Ultra96V2

JTAG / USB Pod

Ultra96V2 Power Supply

SD Card with the Ultra96V2 OOB

Downloads and Installations

Step 1 – Download and install the following at least 1 day prior to the workshop. This may take a significant amount of time and drive space.

Watch the video available [here](#) to show how to configure the installation

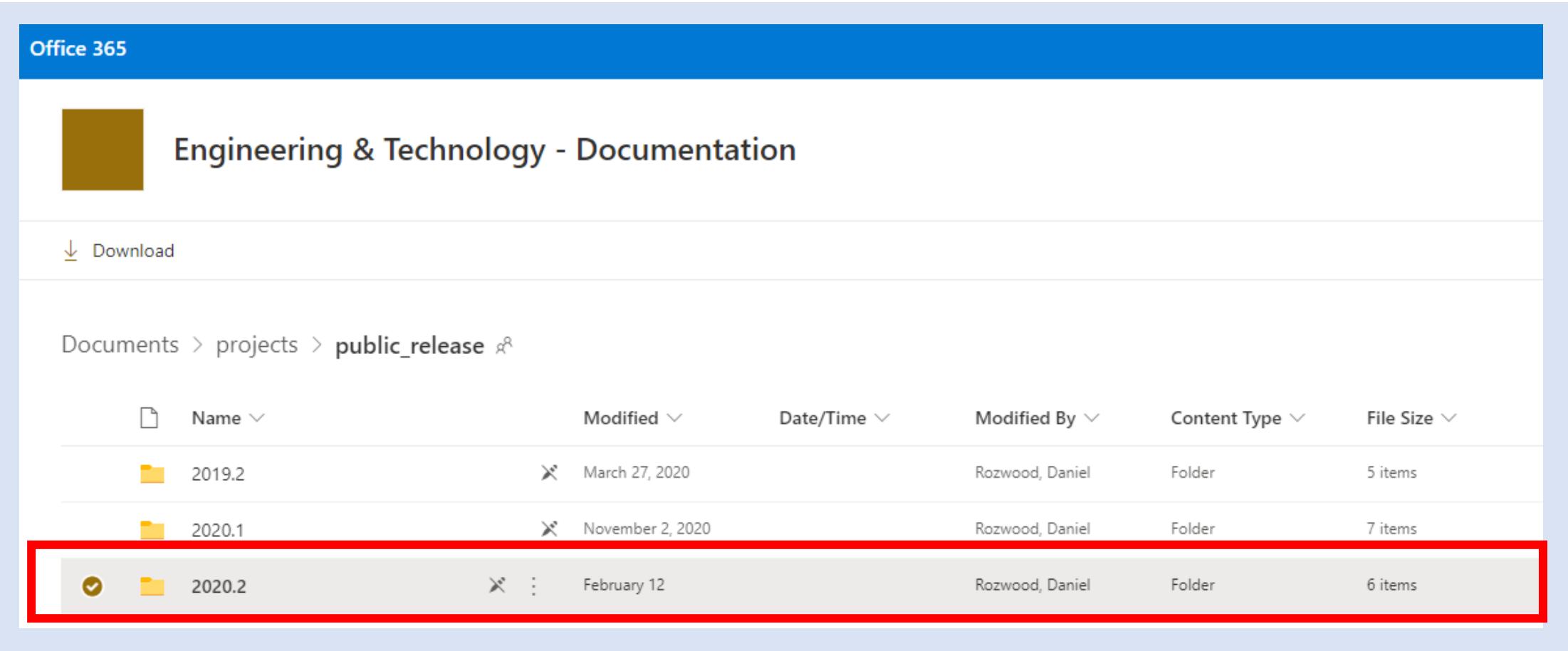
Vitis 2020.2	Download
PetaLinux 2020.2	Download

Lab 3

Creating Embedded Linux Solutions

Lab 3: Creating Embedded Linux Solutions

Step 1 – Click [here](#) and select 2020.2



Office 365

Engineering & Technology - Documentation

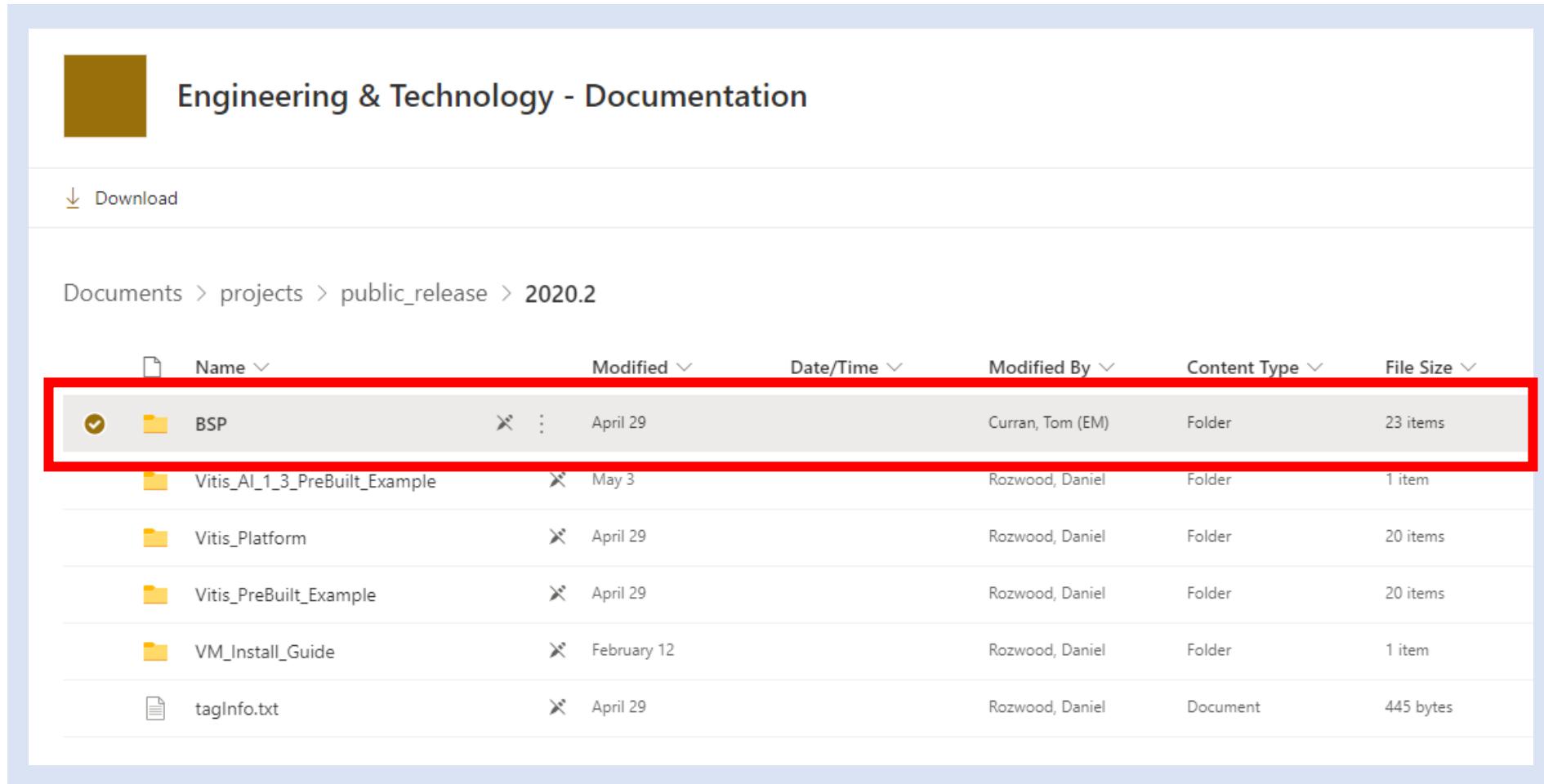
Download

Documents > projects > public_release ↗

Name	Modified	Date/Time	Modified By	Content Type	File Size
2019.2	March 27, 2020		Rozwood, Daniel	Folder	5 items
2020.1	November 2, 2020		Rozwood, Daniel	Folder	7 items
2020.2	February 12		Rozwood, Daniel	Folder	6 items

Lab 3: Creating Embedded Linux Solutions

Step 2 – Select BSP



Engineering & Technology - Documentation

Download

Documents > projects > public_release > 2020.2

Name	Modified	Date/Time	Modified By	Content Type	File Size
BSB	April 29		Curran, Tom (EM)	Folder	23 items
Vitis_AI_1_3_PreBuilt_Example	May 3		Rozwood, Daniel	Folder	1 item
Vitis_Platform	April 29		Rozwood, Daniel	Folder	20 items
Vitis_PreBuilt_Example	April 29		Rozwood, Daniel	Folder	20 items
VM_Install_Guide	February 12		Rozwood, Daniel	Folder	1 item
tagInfo.txt	April 29		Rozwood, Daniel	Document	445 bytes

Lab 3: Creating Embedded Linux Solutions

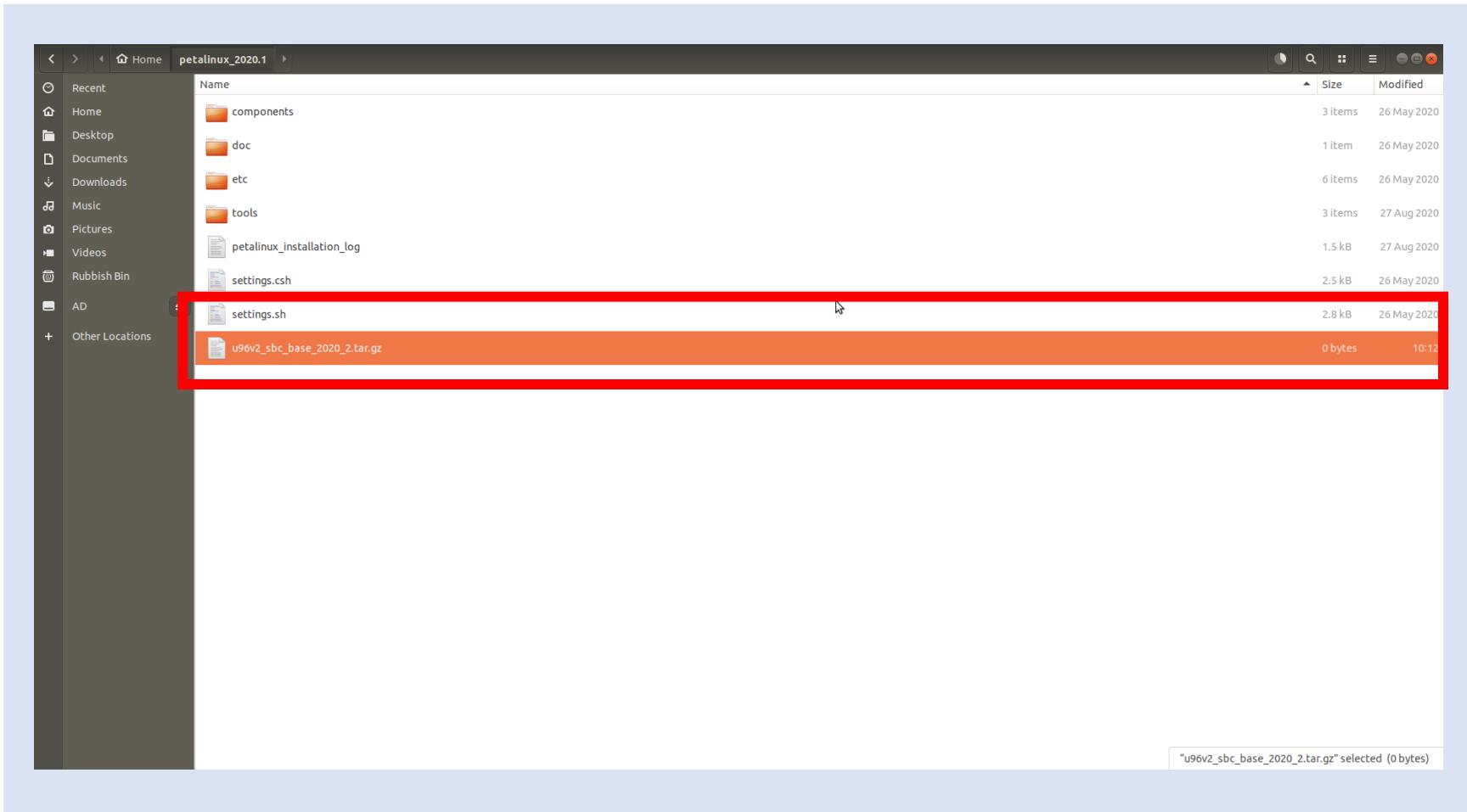
Step 3 – Select and download the Ultra96V2 SBC Base

The screenshot shows a SharePoint document library interface titled "Engineering & Technology - Documentation". The navigation path is "Documents > projects > public_release > 2020.2 > BSP". The list view displays the following files:

Name	Modified	Date/Time	Modified By	Content Type	File Size
pz7010_fmc2_base_2020_2.mds	May 3		Rozwood, Daniel	Document	33 bytes
pz7010_fmc2_base_2020_2.tar.gz	April 26		Rozwood, Daniel	Document	1.02 GB
pz7015_fmc2_base_2020_2.md5	May 3		Rozwood, Daniel	Document	33 bytes
pz7015_fmc2_base_2020_2.tar.gz	April 26		Rozwood, Daniel	Document	1.02 GB
pz7020_fmc2_base_2020_2.md5	May 3		Rozwood, Daniel	Document	33 bytes
pz7020_fmc2_base_2020_2.tar.gz	May 3		Rozwood, Daniel	Document	1.02 GB
pz7030_fmc2_base_2020_2.md5	May 3		Rozwood, Daniel	Document	33 bytes
pz7030_fmc2_base_2020_2.tar.gz	May 3		Rozwood, Daniel	Document	1.02 GB
u96v2_sbc_base_2020_2.mds	May 3		Rozwood, Daniel	Document	33 bytes
u96v2_sbc_base_2020_2.tar.gz	April 27		Rozwood, Daniel	Document	4.90 GB

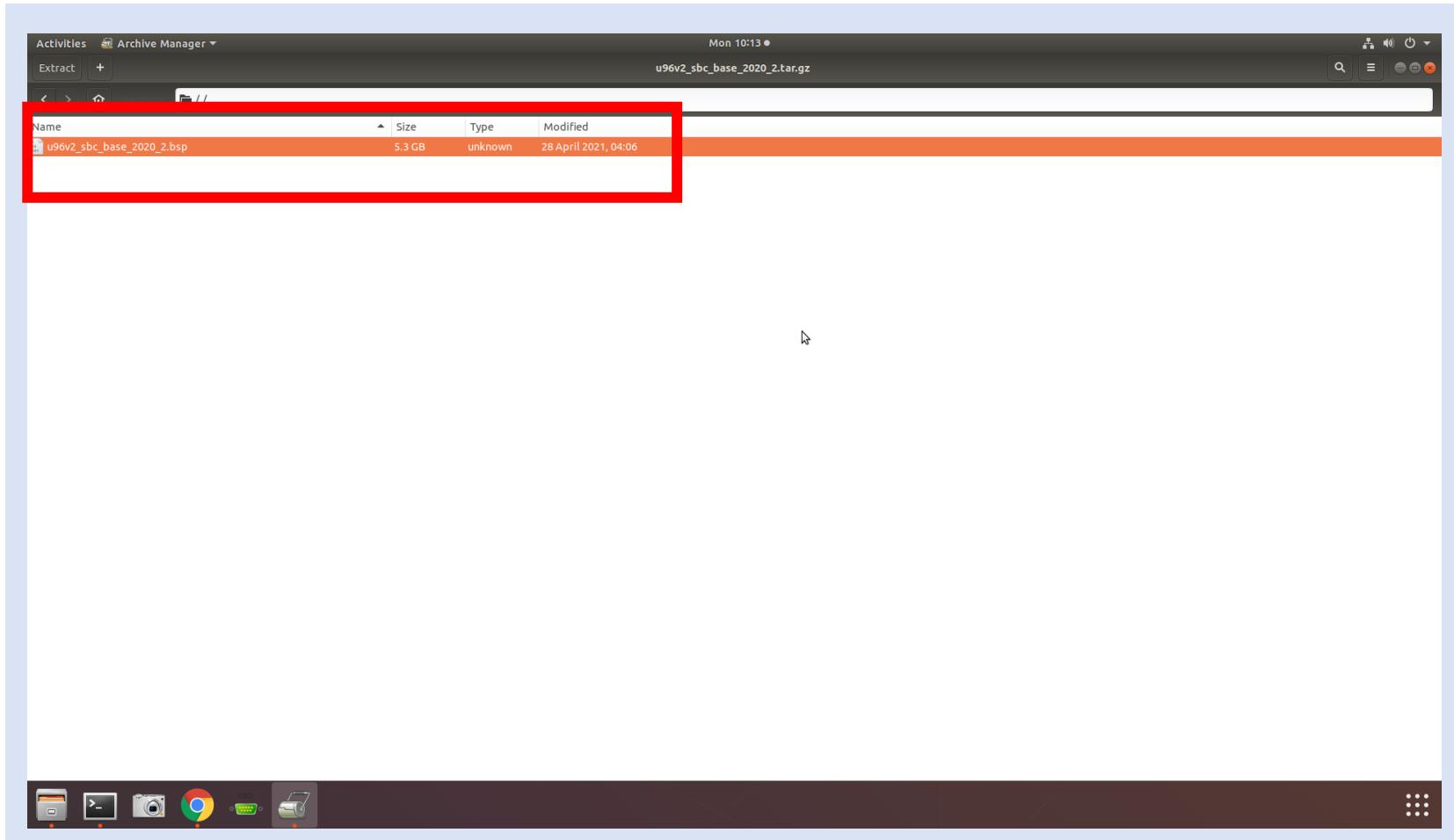
Lab 3: Creating Embedded Linux Solutions

Step 4 – Double click on the Ultra96V2 tar.gz to open the archive



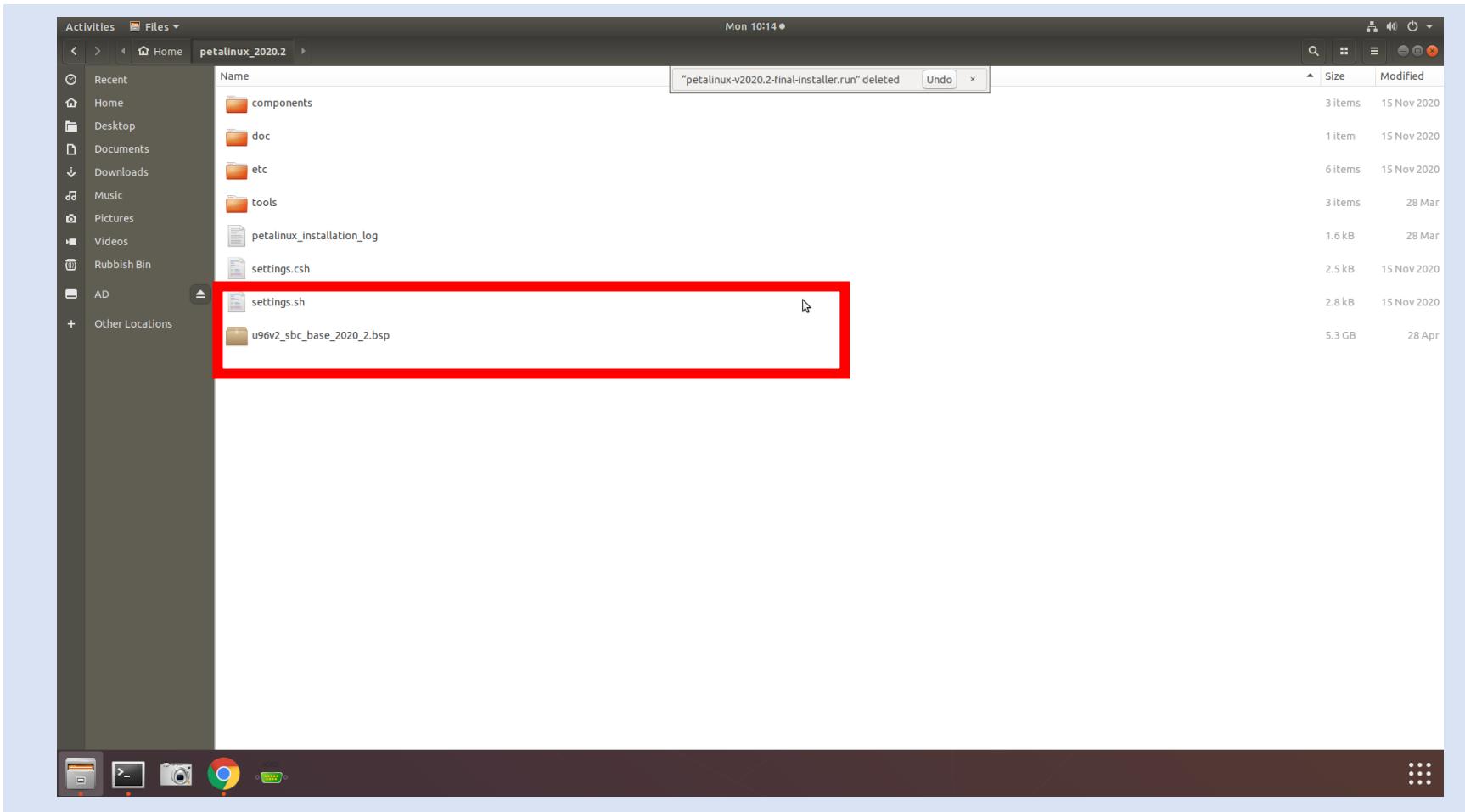
Lab 3: Creating Embedded Linux Solutions

Step 5 – Extract the BSP to the Petalinux directory



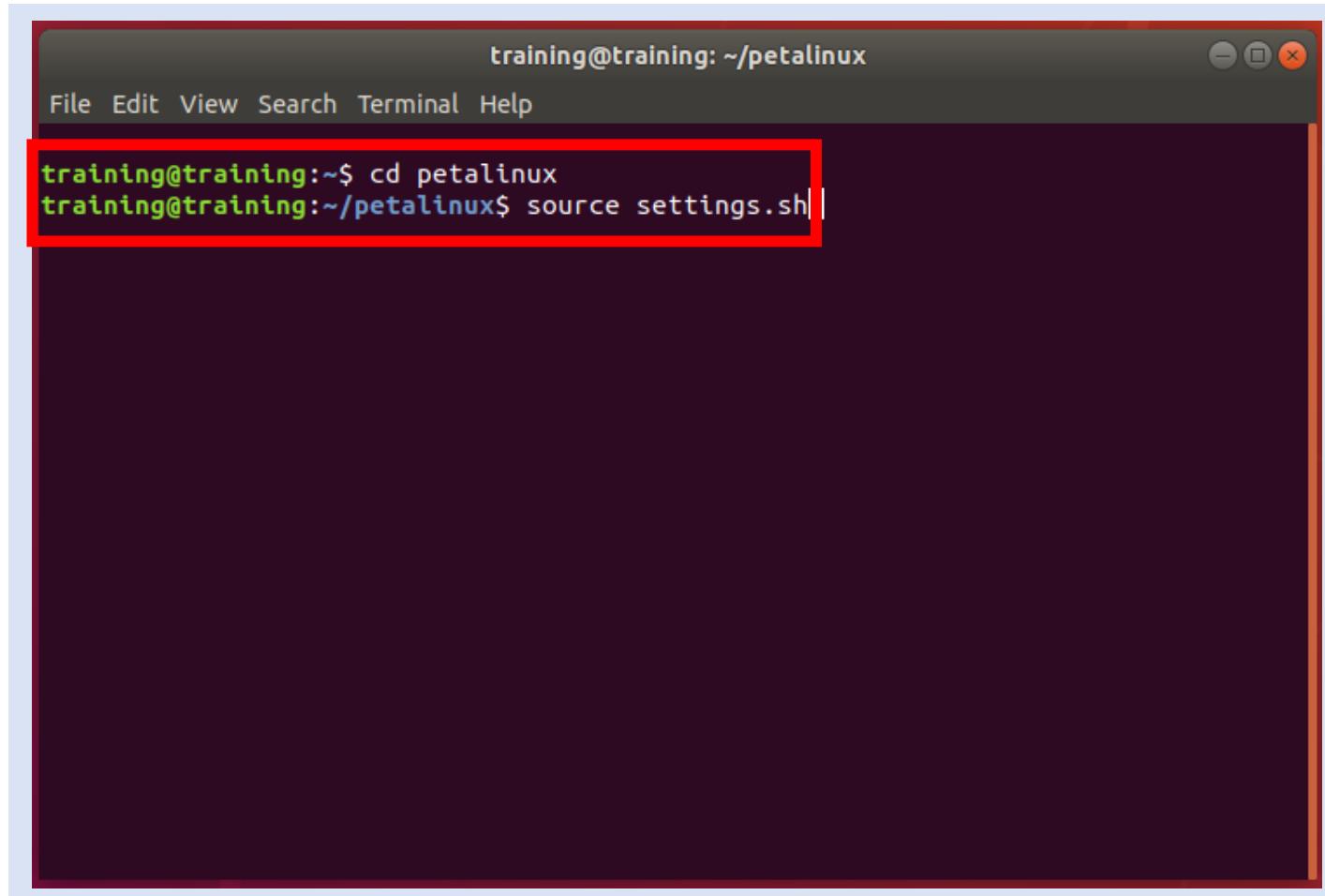
Lab 3: Creating Embedded Linux Solutions

Step 6 – Delete the compressed file, you should see the BSP in the directory



Lab 3: Creating Embedded Linux Solutions

Step 7 – In the petalinux directory, run the command *source settings.sh*



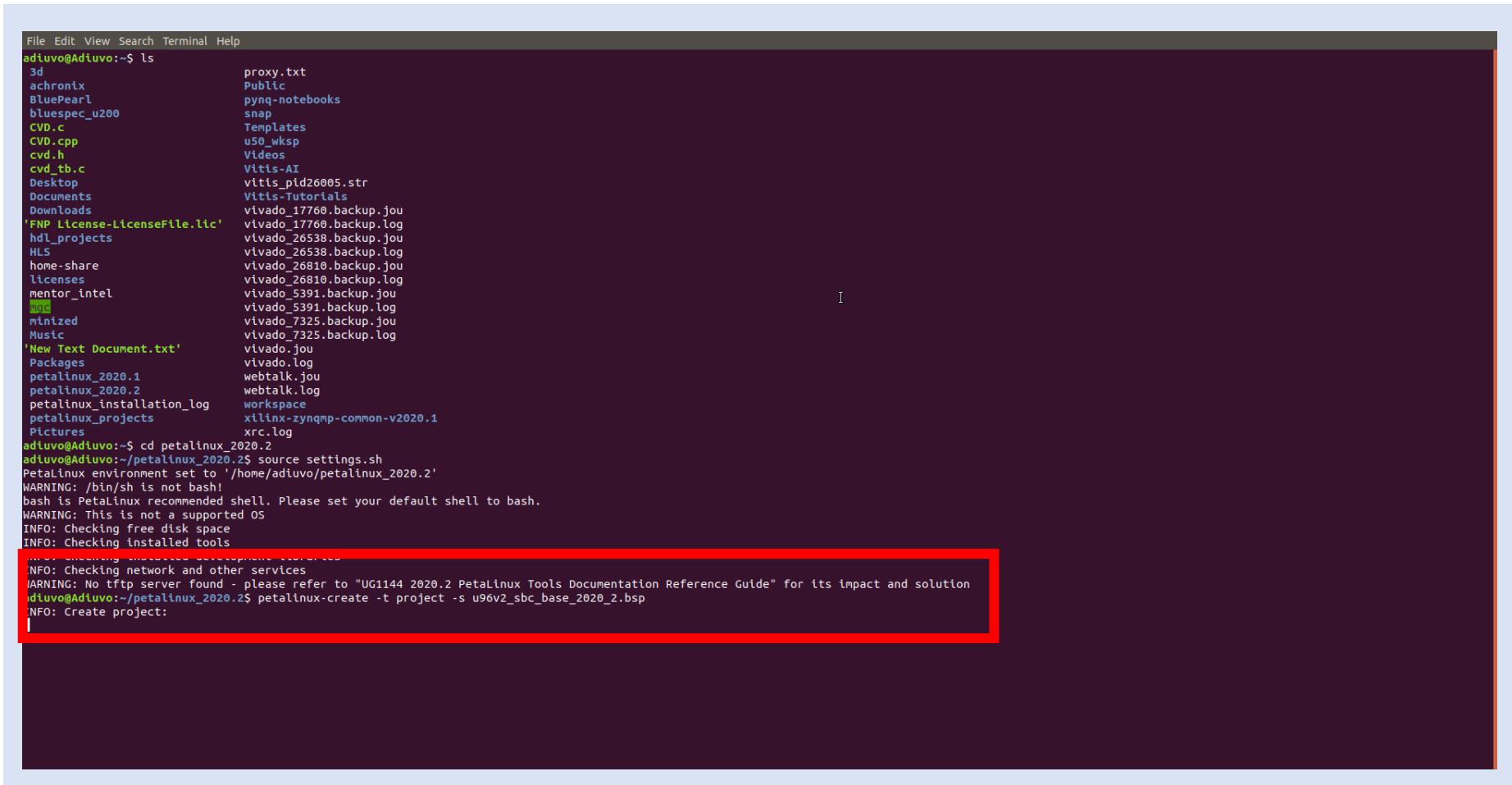
A screenshot of a terminal window titled "training@training: ~/petalinux". The window has a standard Linux-style title bar with icons for minimize, maximize, and close. Below the title bar is a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The main terminal area shows two commands in green text:

```
training@training:~$ cd petalinux
training@training:~/petalinux$ source settings.sh
```

The last command, "source settings.sh", is highlighted with a red rectangular box.

Lab 3: Creating Embedded Linux Solutions

Step 8 – Create a new project, using the command *petalinux-create -t project -s <BSP Name>*

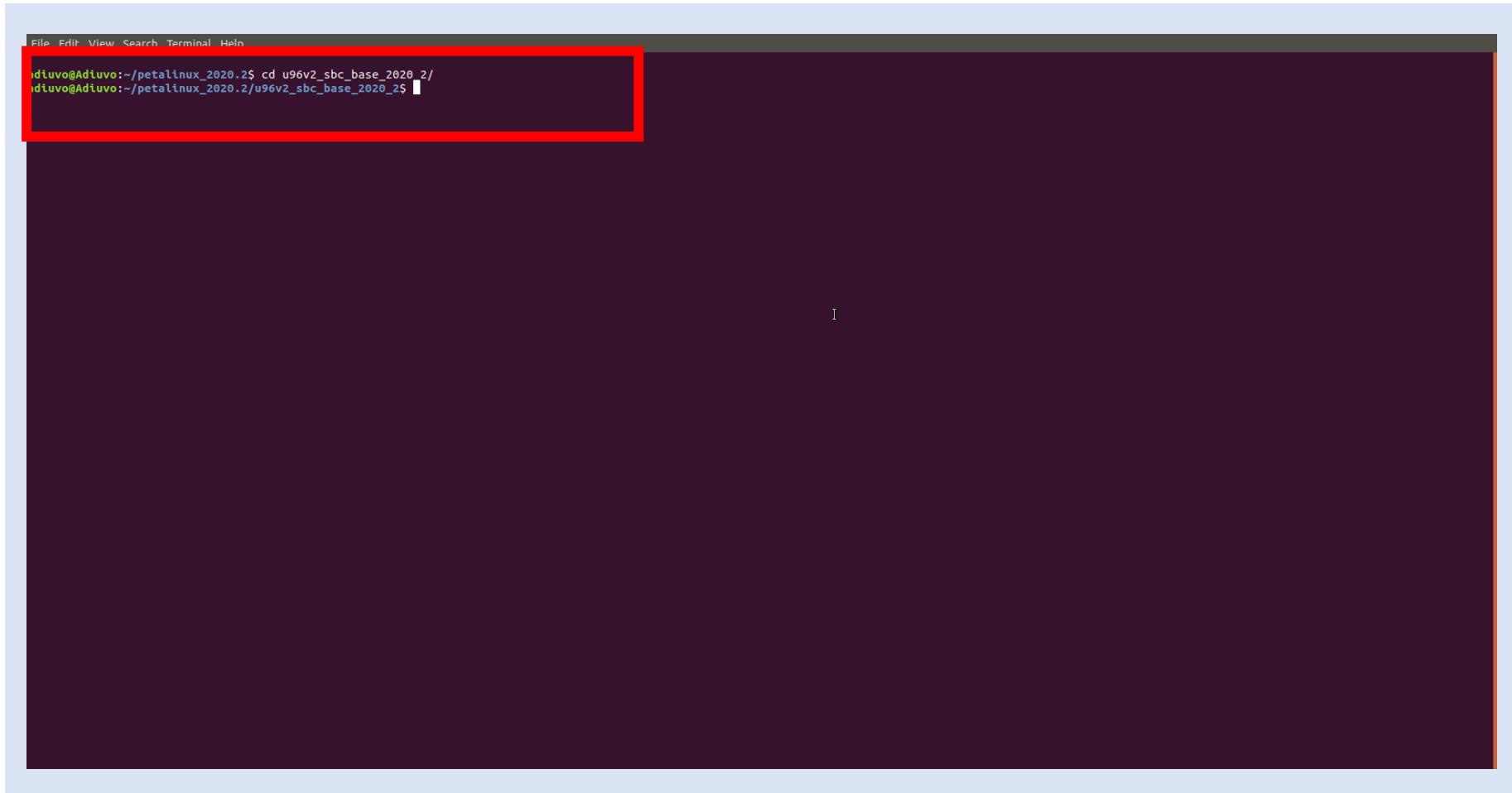


The screenshot shows a terminal window with a dark background and light-colored text. At the top, there's a menu bar with options: File, Edit, View, Search, Terminal, Help. Below the menu, the terminal prompt is `adtuvo@Adtuvo:`. The user runs the command `petalinux-create -t project -s u96v2_sbc_base_2020_2.bsp`. A red rectangular box highlights the output of the command, which includes several informational messages: "INFO: Create project:", "INFO: Checking installed tools", "INFO: Checking network and other services", and "WARNING: No tftp server found - please refer to "UG1144 2020.2 PetaLinux Tools Documentation Reference Guide" for its impact and solution". The rest of the terminal window shows a list of files and directories in the current directory, including ".git", "LICENSE", "README.md", "Vitis-AI", "Vitis-Tutorials", and various Vivado backup files.

```
File Edit View Search Terminal Help
adtuvo@Adtuvo:~$ ls
3d           proxy.txt
achronix      Public
BluePearl     pynq-notebooks
bluespec_u200 snap
CV0.c         Templates
CV0.cpp       u50_wksp
cvd.h         Videos
cvd_tb.c     Vitis-AI
Desktop      vitis_pid26005.str
Documents    Vitis-Tutorials
Downloads   vivado_17760.backup.jou
'FNP_License-LicenseFile.lic' vivado_17760.backup.log
hdl_projects vivado_26538.backup.jou
HLS          vivado_26538.backup.log
home-share   vivado_26810.backup.jou
licenses     vivado_26810.backup.log
mentor_intel vivado_5391.backup.jou
'.git'        vivado_5391.backup.log
mntzized     vivado_7325.backup.jou
Music        vivado_7325.backup.log
'New Text Document.txt' vivado.jou
Packages     vivado.log
petalinux_2020.1 webtalk.jou
petalinux_2020.2 webtalk.log
petalinux_installation_log workspace
petalinux_projects xilinx-zynqmp-common-v2020.1
Pictures     xrc.log
adtuvo@Adtuvo:~$ cd petalinux_2020.2
adtuvo@Adtuvo:~/petalinux_2020.2$ source settings.sh
Petalinux environment set to '/home/adtuvo/petalinux_2020.2'
WARNING: /bin/sh is not bash!
bash is Petalinux recommended shell. Please set your default shell to bash.
WARNING: This is not a supported OS
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
WARNING: No tftp server found - please refer to "UG1144 2020.2 PetaLinux Tools Documentation Reference Guide" for its impact and solution
adtuvo@Adtuvo:~/petalinux_2020.2$ petalinux-create -t project -s u96v2_sbc_base_2020_2.bsp
INFO: Create project:
```

Lab 3: Creating Embedded Linux Solutions

Step 9 – Change Directory into the project directory just created

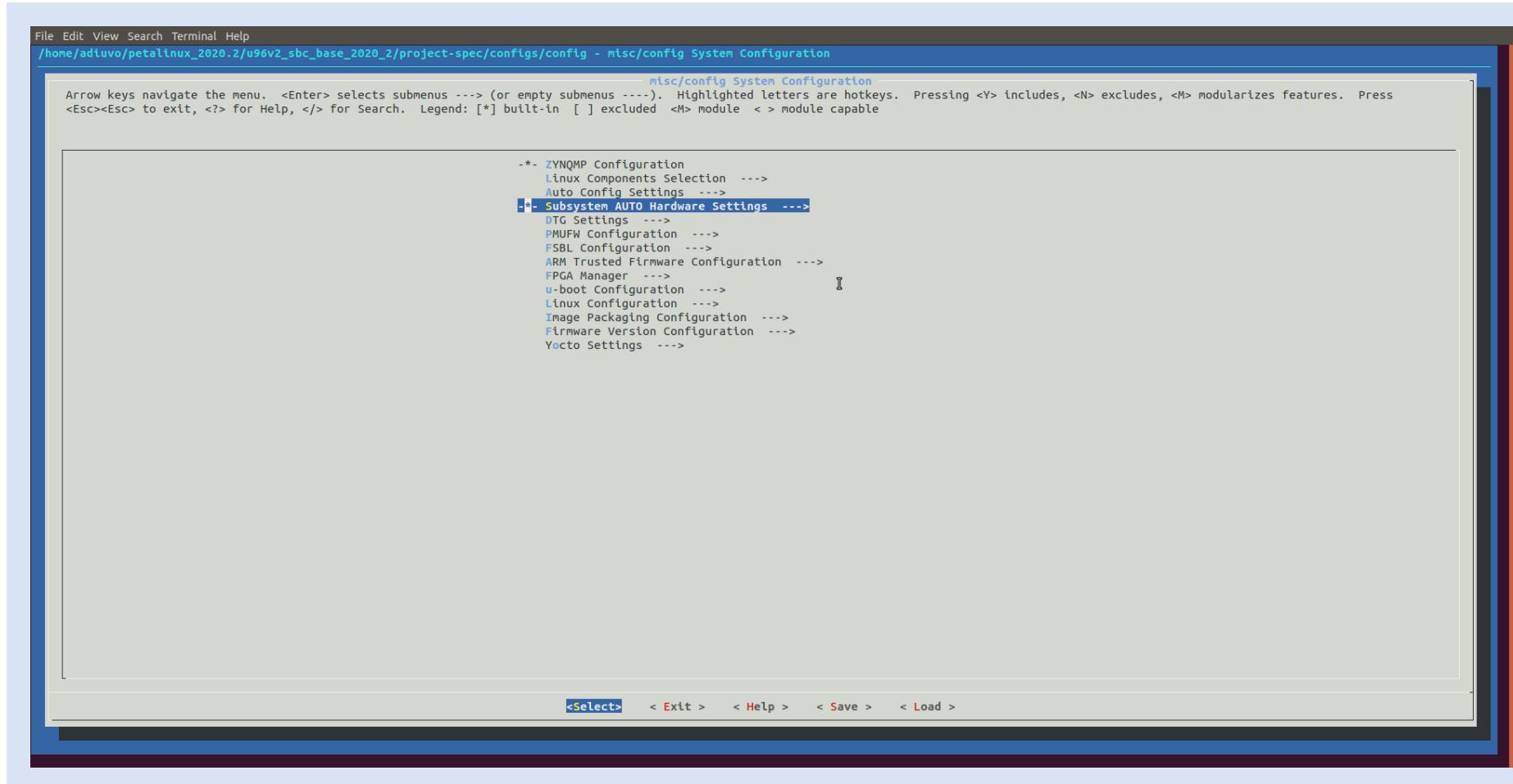


```
File Edit View Search Terminal Help
adiuvo@Adiuvo:~/petalinux_2020.2$ cd u96v2_sbc_base_2020_2/
adiuvo@Adiuvo:~/petalinux_2020.2/u96v2_sbc_base_2020_2$
```

A screenshot of a terminal window titled "Terminal". The window has a dark background and a light blue header bar at the top. In the header bar, there are menu options: File, Edit, View, Search, Terminal, and Help. Below the header, the terminal prompt shows the user's name "adiuvo" followed by the path "Adiuvo:~/petalinux_2020.2\$". The user then enters the command "cd u96v2_sbc_base_2020_2/" and presses Enter. The terminal then displays the path again, followed by a dollar sign and a small black square icon. A red rectangular box highlights the command "cd u96v2_sbc_base_2020_2/".

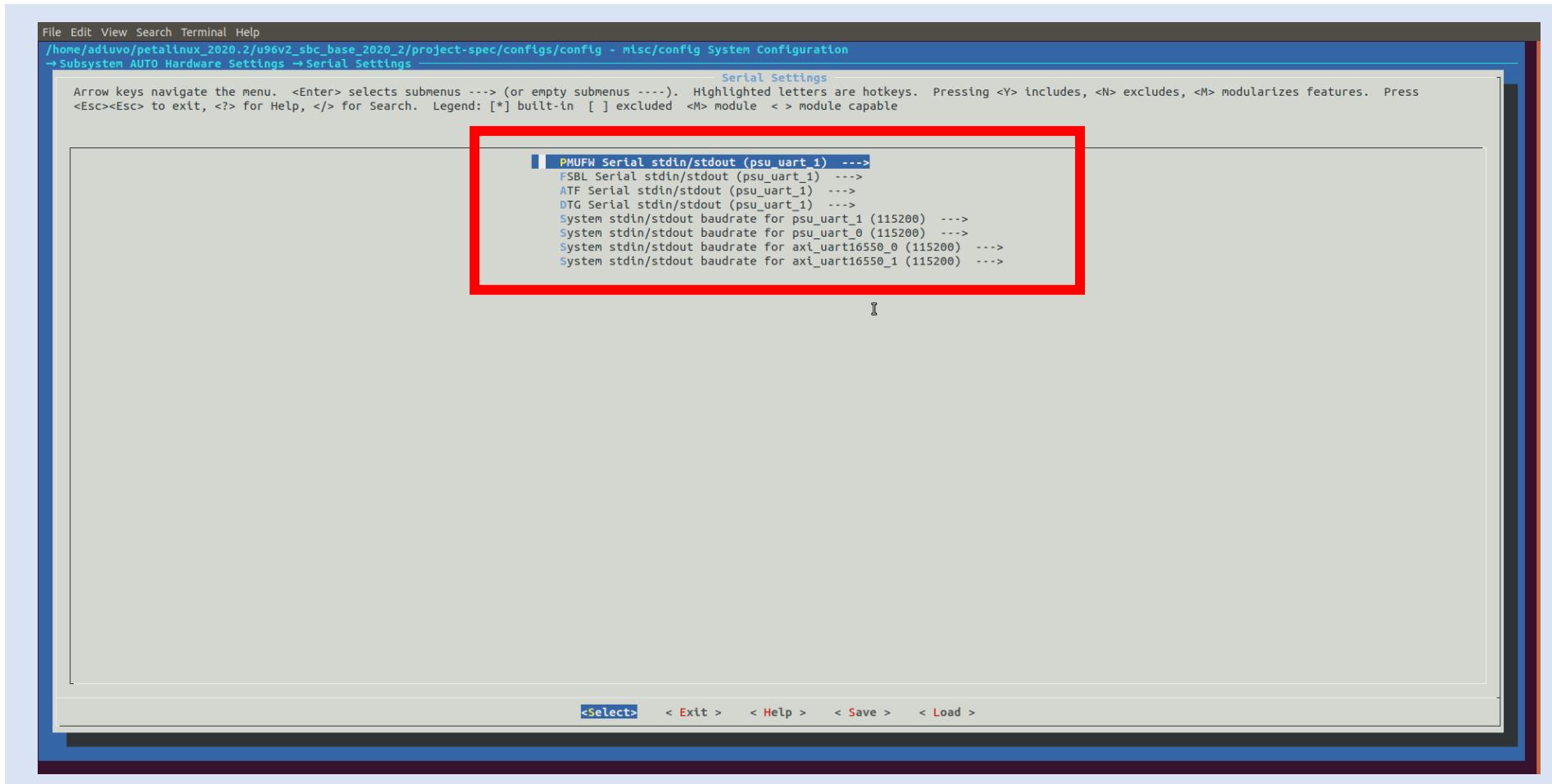
Lab 3: Creating Embedded Linux Solutions

Step 10 – In the terminal issue the command `petalinux-config` This will open the petalinux project configuration dialog. Scroll down to Subsystem AUTO Hardware Settings press enter



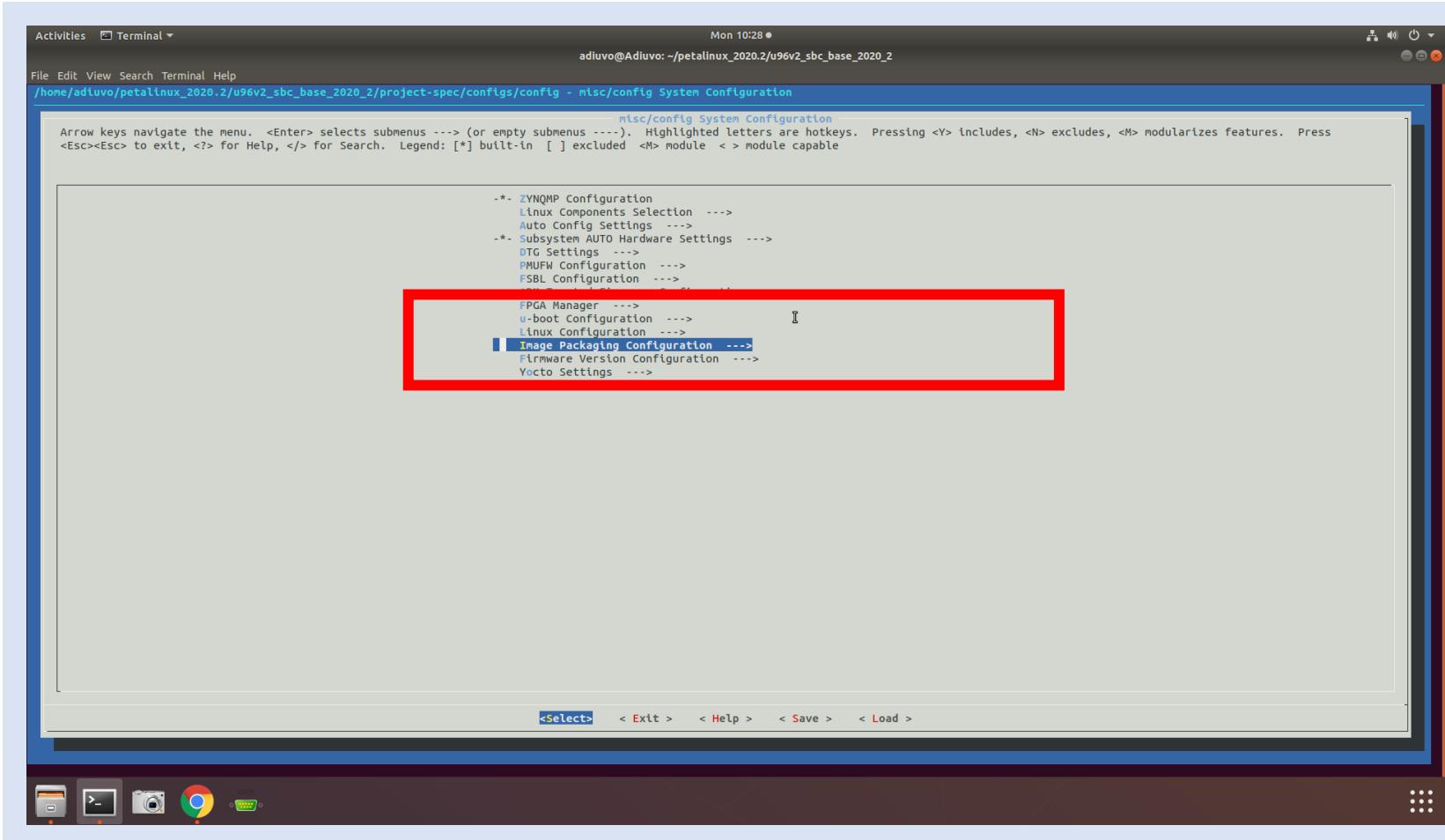
Lab 3: Creating Embedded Linux Solutions

Step 11 – Select the serial port and check that all of them are PSU_UART1



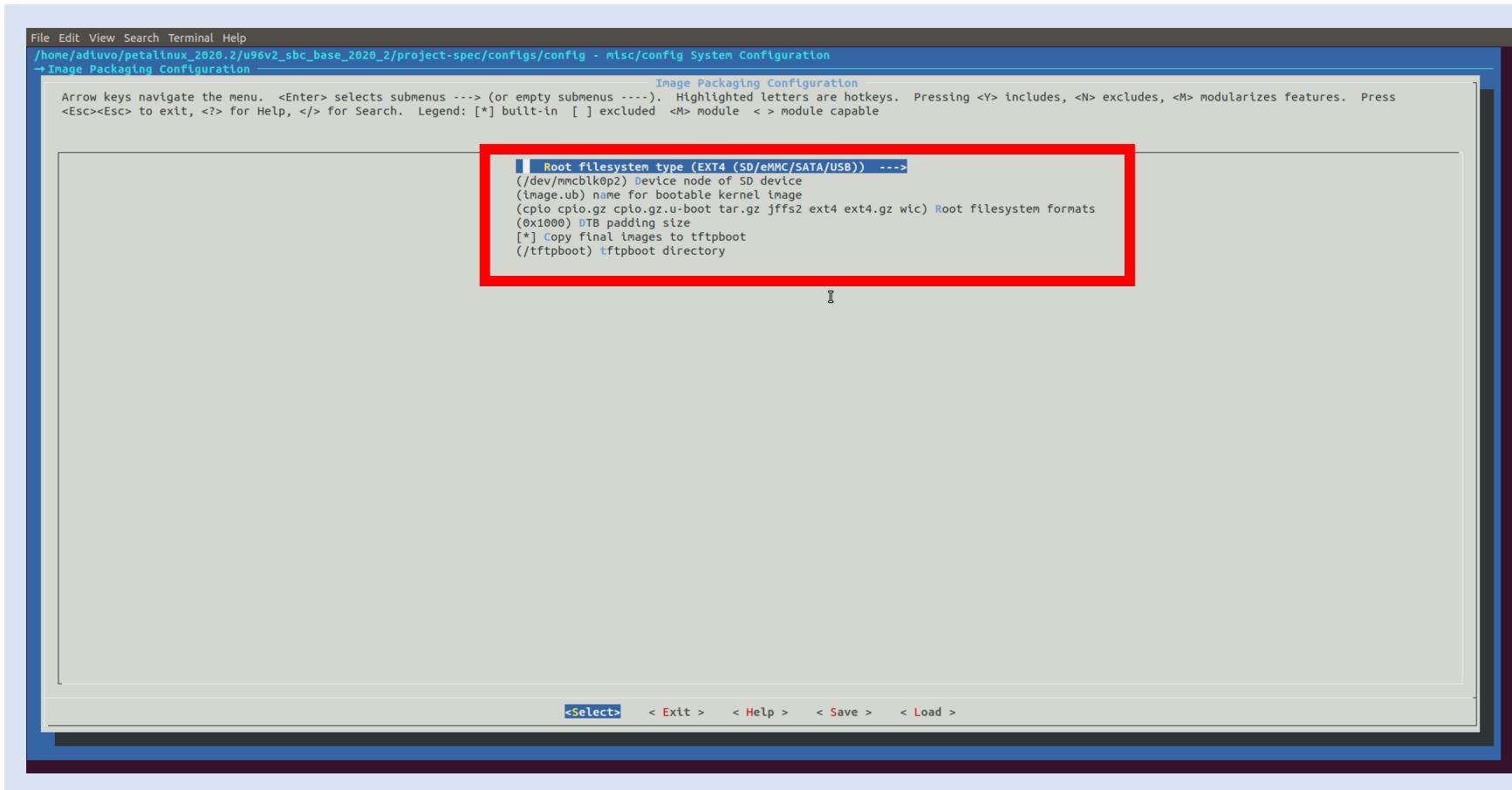
Lab 3: Creating Embedded Linux Solutions

Step 12 – Navigate back to main and select Image Packaging Configuration



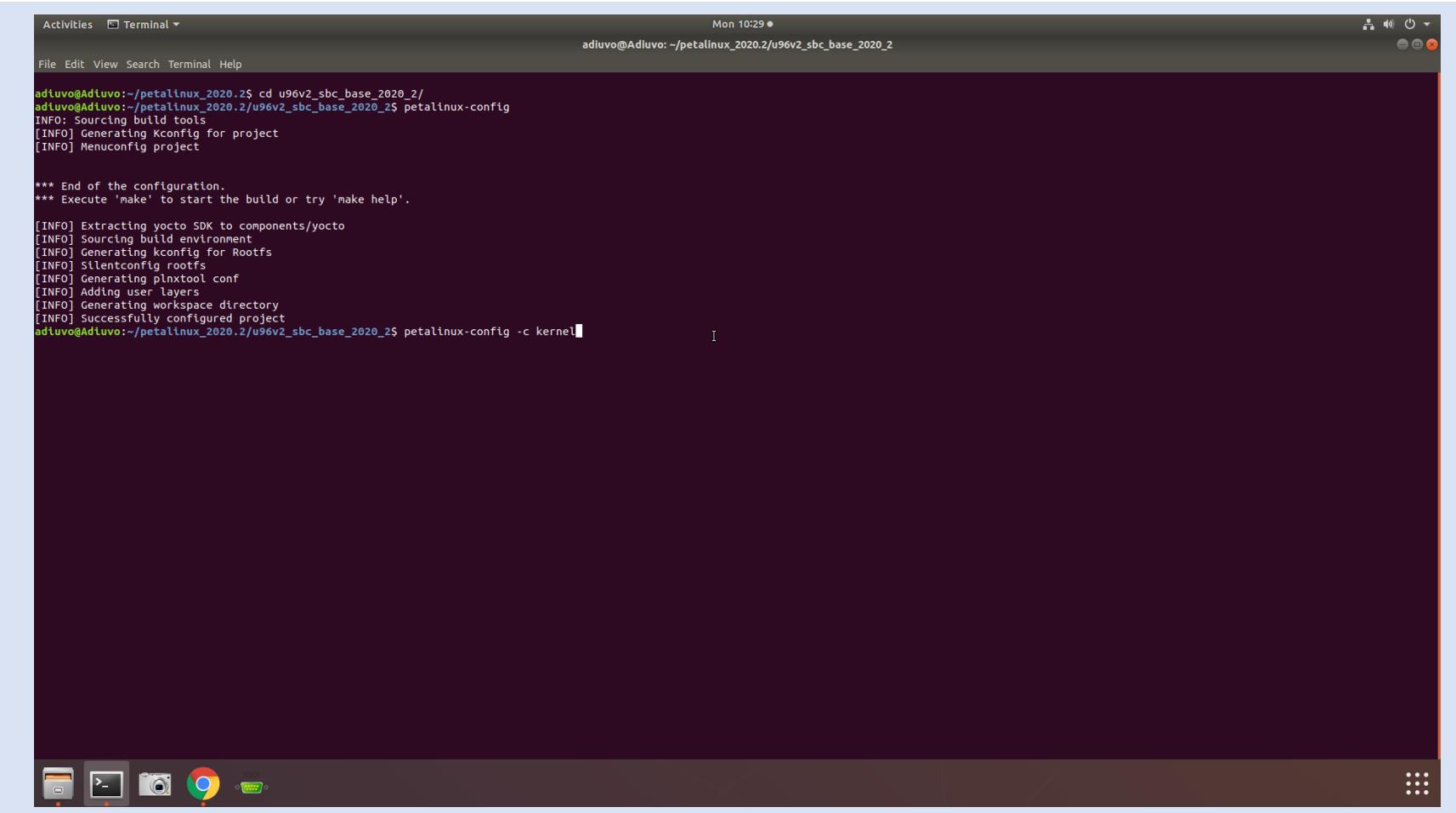
Lab 3: Creating Embedded Linux Solutions

Step 13 – Notice the SD Card is used to store the Root File System – Exit the menu and do not save any changes.



Lab 3: Creating Embedded Linux Solutions

Step 14 – Enter the command *petalinux-config –c kernel*



The screenshot shows a terminal window titled "Terminal" with the command "petalinux-config -c kernel" being run. The terminal output indicates the configuration process is starting, including sourcing build tools, generating Kconfig and menuconfig files, extracting Yocto SDK components, and generating Rootfs and pnxtool conf files. The project is successfully configured.

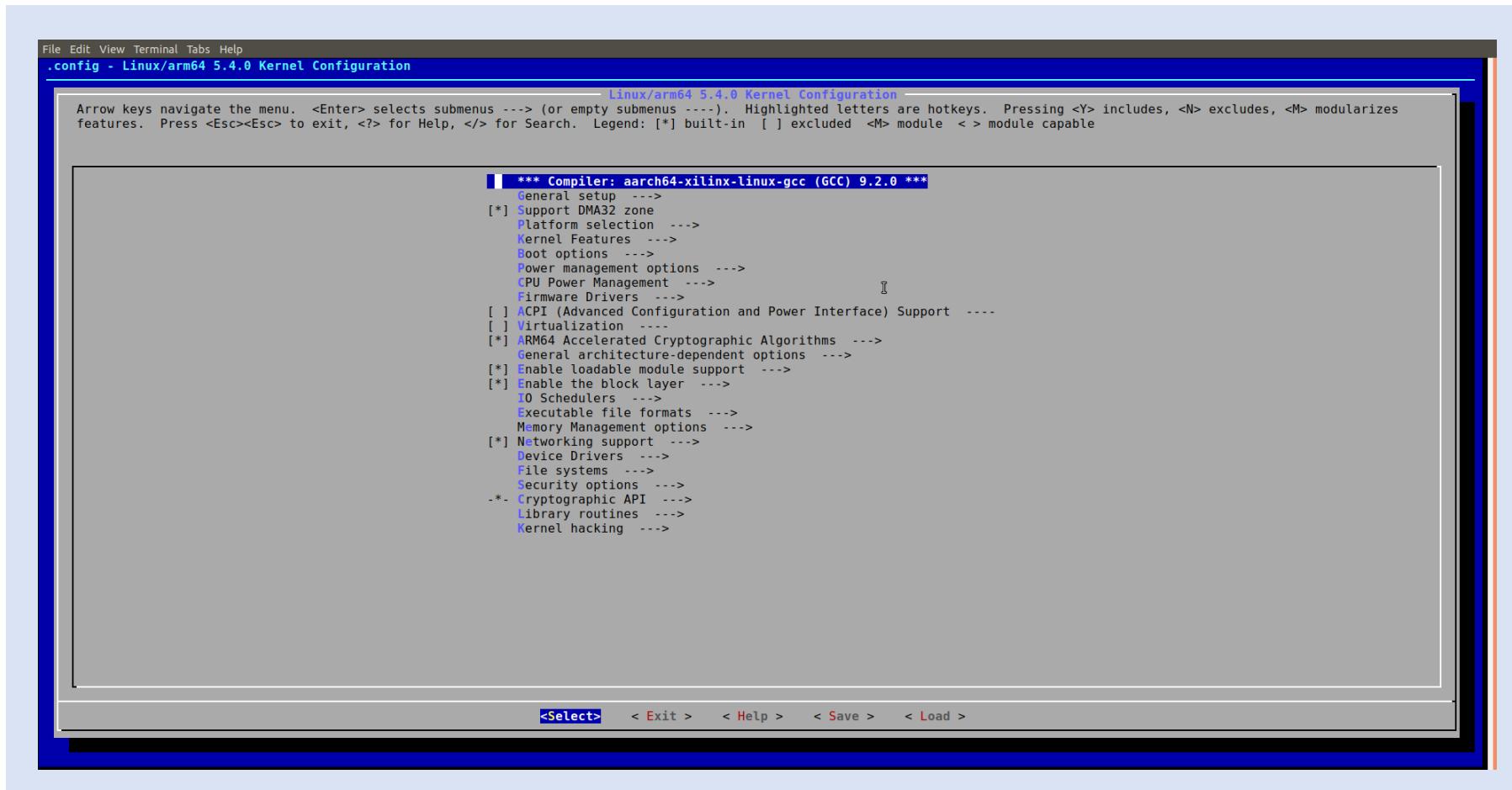
```
Activities Terminal Mon 10:29 ● adiuvvo@Aduvo: ~/petalinux_2020.2/u96v2_sbc_base_2020_2
File Edit View Search Terminal Help
adiuvvo@Aduvo:~/petalinux_2020.2$ cd u96v2_sbc_base_2020_2/
adiuvvo@Aduvo:~/petalinux_2020.2/u96v2_sbc_base_2020_2$ petalinux-config
INFO: Sourcing build tools
[INFO] Generating Kconfig for project
[INFO] Menuconfig project

*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.

[INFO] Extracting yocto SDK to components/yocto
[INFO] Sourcing build environment
[INFO] Generating kconfig for Rootfs
[INFO] Silentconfig rootfs
[INFO] Generating pnxtool conf
[INFO] Adding user layers
[INFO] Generating workspace directory
[INFO] Successfully configured project
adiuvvo@Aduvo:~/petalinux_2020.2/u96v2_sbc_base_2020_2$ petalinux-config -c kernel
```

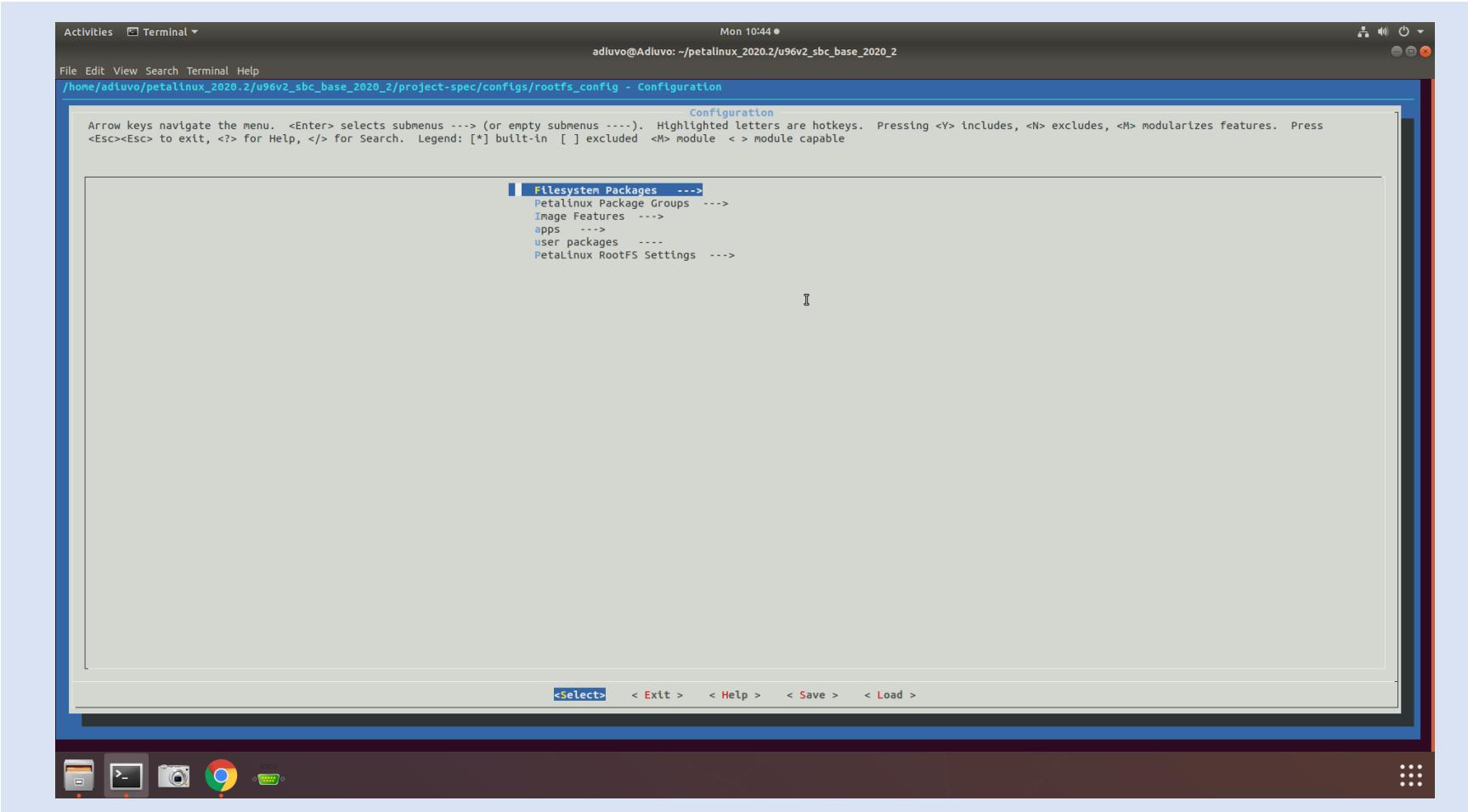
Lab 3: Creating Embedded Linux Solutions

Step 15 – This is the Kernel Configuration, explore the settings and close the menu without making any changes.



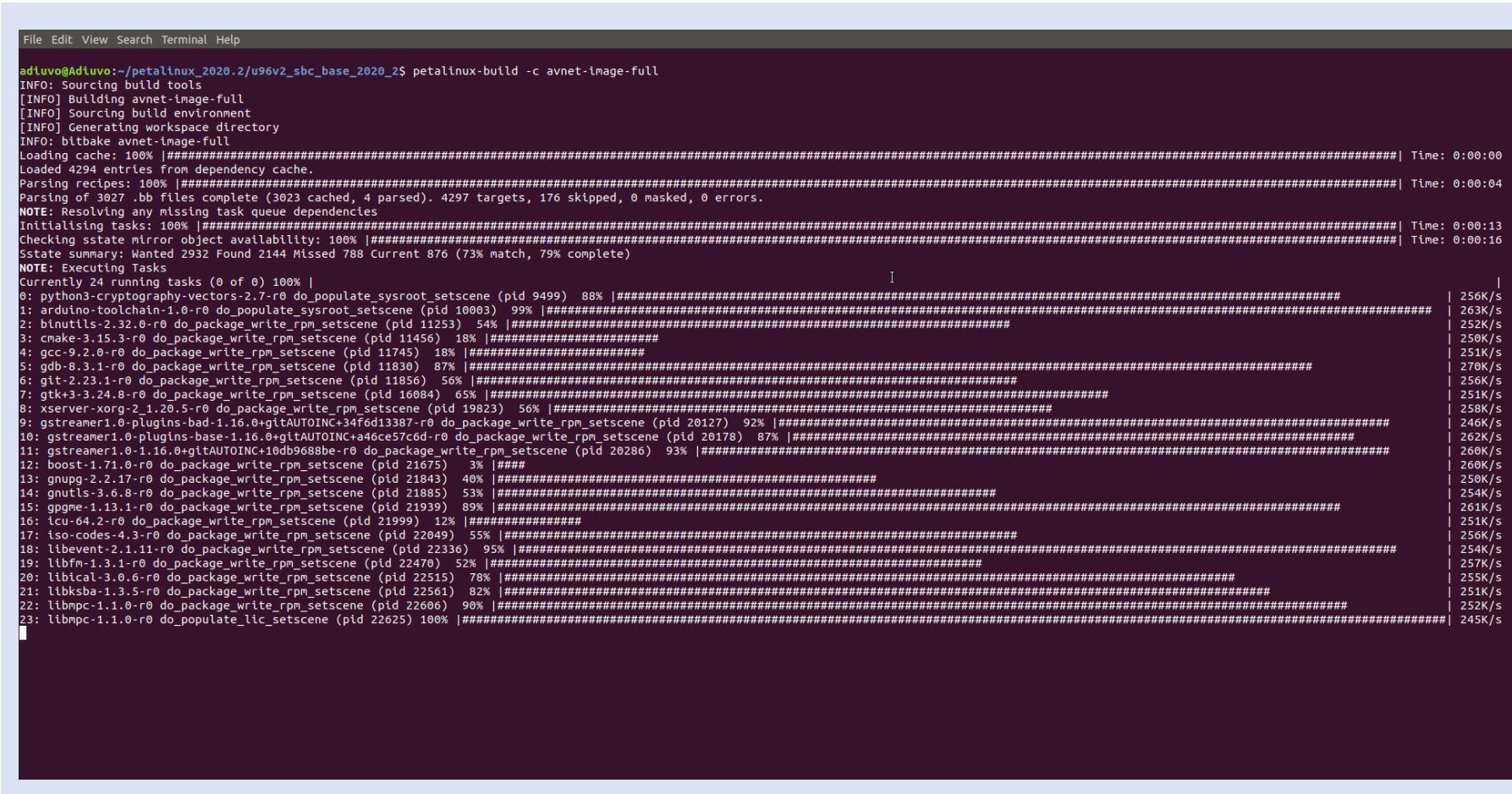
Lab 3: Creating Embedded Linux Solutions

Step 16 – In the terminal enter the command `petalinux-config -c rootfs` This is the Kernel Configuration, explore the settings and close the menu without making any changes.



Lab 3: Creating Embedded Linux Solutions

Step 17 – Build the image by typing *petalinux-build -c avnet-image-full*

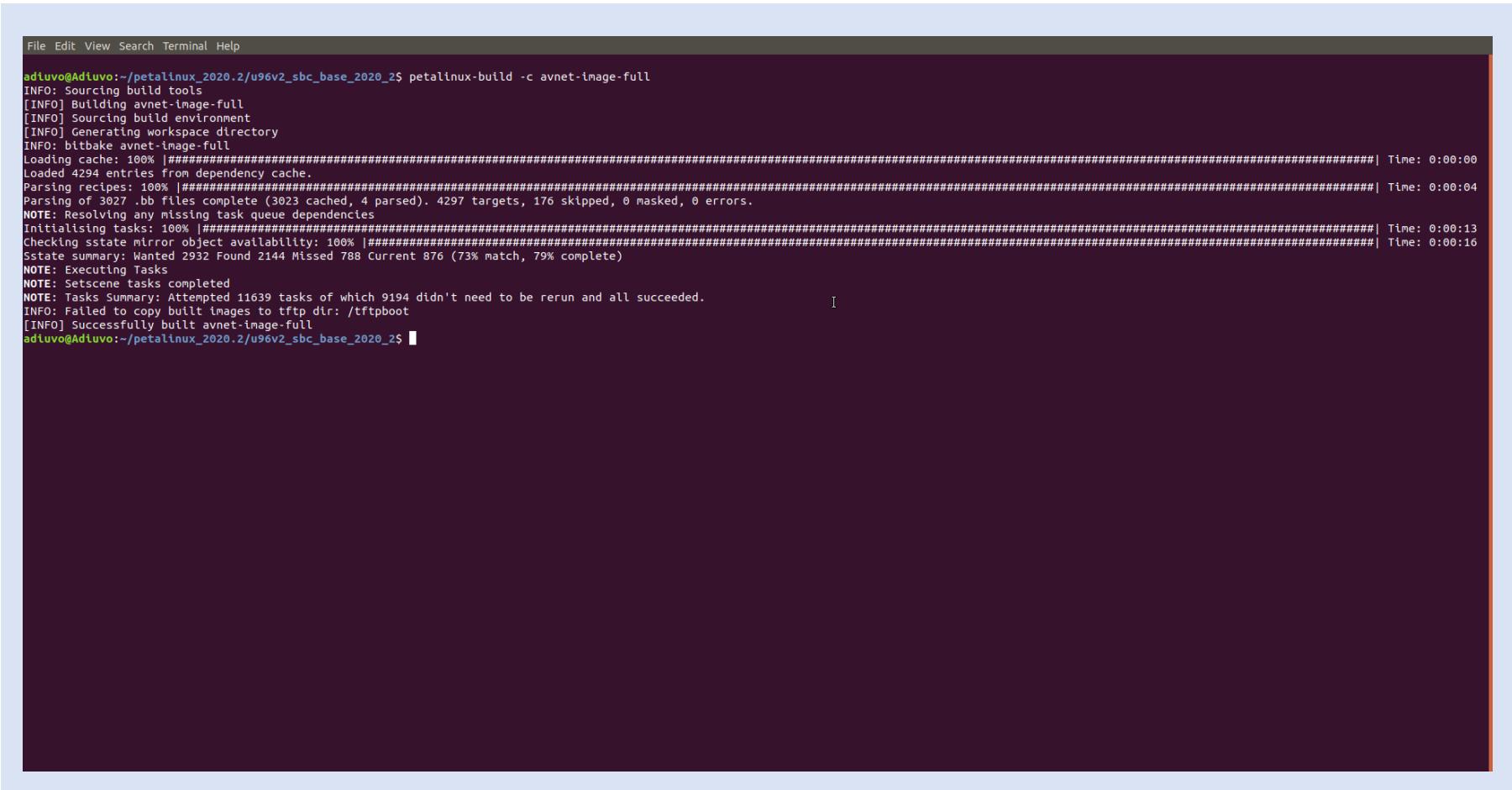


A terminal window showing the output of the command `petalinux-build -c avnet-image-full`. The window has a dark background and light-colored text. The text shows the build process starting with sourcing build tools, generating workspace, and then using bitbake to build the image. It displays the progress of 24 tasks, each with a PID, name, and percentage complete. The tasks include various packages like python3-cryptography, arduino-toolchain, binutils, cmake, gcc, gdb, gtk+, xserver-xorg, gstreamer, boost, gnupg, gnutls, gpgme, lcu, lso-codes, libevent, libfm, libical, libksba, libmpc, and libmpc_lic. The build is progressing at approximately 250K/s.

```
File Edit View Search Terminal Help  
adiuvo@Adiuv0:~/petalinux_2020.2/u96v2_sbc_base_2020_2$ petalinux-build -c avnet-image-full  
INFO: Sourcing build tools  
[INFO] Building avnet-image-full  
[INFO] Sourcing build environment  
[INFO] Generating workspace directory  
INFO: bitbake avnet-image-full  
Loading cache: 100% [#####################################################################] Time: 0:00:00  
Loaded 4294 entries from dependency cache.  
Parsing recipes: 100% [#####################################################################] Time: 0:00:04  
Parsing of 3827 .bb files complete (3023 cached, 4 parsed). 4297 targets, 176 skipped, 0 masked, 0 errors.  
NOTE: Resolving any missing task queue dependencies  
Initialising tasks: 100% [#####################################################################] Time: 0:00:13  
Checking sstate mirror object availability: 100% [#####################################################################] Time: 0:00:16  
Sstate summary: Wanted 2932 Found 2144 Missed 788 Current 876 (73% match, 79% complete)  
NOTE: Executing Tasks  
Currently 24 running tasks (0 of 0) 100% |  
0: python3-cryptography-vectors-2.7-r0 do_populate_sysroot_setscene (pid 9499) 88% [#####
1: arduino-toolchain-1.0-r0 do_populate_sysroot_setscene (pid 10003) 99% [#####
2: binutils-2.32.0-r0 do_package_write_rpm_setscene (pid 11253) 54% [#####
3: cmake-3.15.3-r0 do_package_write_rpm_setscene (pid 11456) 18% [#####
4: gcc-9.2.0-r0 do_package_write_rpm_setscene (pid 11745) 18% [#####
5: gdb-8.3.1-r0 do_package_write_rpm_setscene (pid 11830) 87% [#####
6: gtk+-2.23.1-r0 do_package_write_rpm_setscene (pid 11856) 56% [#####
7: gtk+-3.24.8-r0 do_package_write_rpm_setscene (pid 16084) 65% [#####
8: xserver-xorg-2.1.20.5-r0 do_package_write_rpm_setscene (pid 19823) 56% [#####
9: gstreamer1.0-plugins-bad-1.16.0+gitAUTOINC+34f6d13387-r0 do_package_write_rpm_setscene (pid 20127) 92% [#####
10: gstreamer1.0-plugins-base-1.16.0+gitAUTOINC+a46ce57c6d-r0 do_package_write_rpm_setscene (pid 20178) 87% [#####
11: gstreamer1.0-1.16.0+gitAUTOINC+10db9688be-r0 do_package_write_rpm_setscene (pid 20286) 93% [#####
12: boost-1.71.0-r0 do_package_write_rpm_setscene (pid 21675) 3% [#####
13: gnupg-2.2.17-r0 do_package_write_rpm_setscene (pid 21843) 40% [#####
14: gnutls-3.6.8-r0 do_package_write_rpm_setscene (pid 21885) 53% [#####
15: gpgme-1.13.1-r0 do_package_write_rpm_setscene (pid 21939) 89% [#####
16: lcu-64.2-r0 do_package_write_rpm_setscene (pid 21999) 12% [#####
17: lso-codes-4.3-r0 do_package_write_rpm_setscene (pid 22049) 55% [#####
18: libevent-2.1.11-r0 do_package_write_rpm_setscene (pid 22336) 95% [#####
19: libfm-1.3.1-r0 do_package_write_rpm_setscene (pid 22470) 52% [#####
20: libical-3.0.6-r0 do_package_write_rpm_setscene (pid 22515) 78% [#####
21: libksba-1.3.5-r0 do_package_write_rpm_setscene (pid 22561) 82% [#####
22: libmpc-1.1.0-r0 do_package_write_rpm_setscene (pid 22606) 90% [#####
23: libmpc-1.1.0-r0 do_populate_lic_setscene (pid 22625) 100% [#####
| 250K/s  
| 263K/s  
| 252K/s  
| 250K/s  
| 251K/s  
| 270K/s  
| 256K/s  
| 251K/s  
| 258K/s  
| 246K/s  
| 262K/s  
| 260K/s  
| 260K/s  
| 250K/s  
| 254K/s  
| 261K/s  
| 251K/s  
| 256K/s  
| 254K/s  
| 257K/s  
| 255K/s  
| 251K/s  
| 252K/s  
| 245K/s
```

Lab 3: Creating Embedded Linux Solutions

Step 18 – Wait for the image to complete its build.

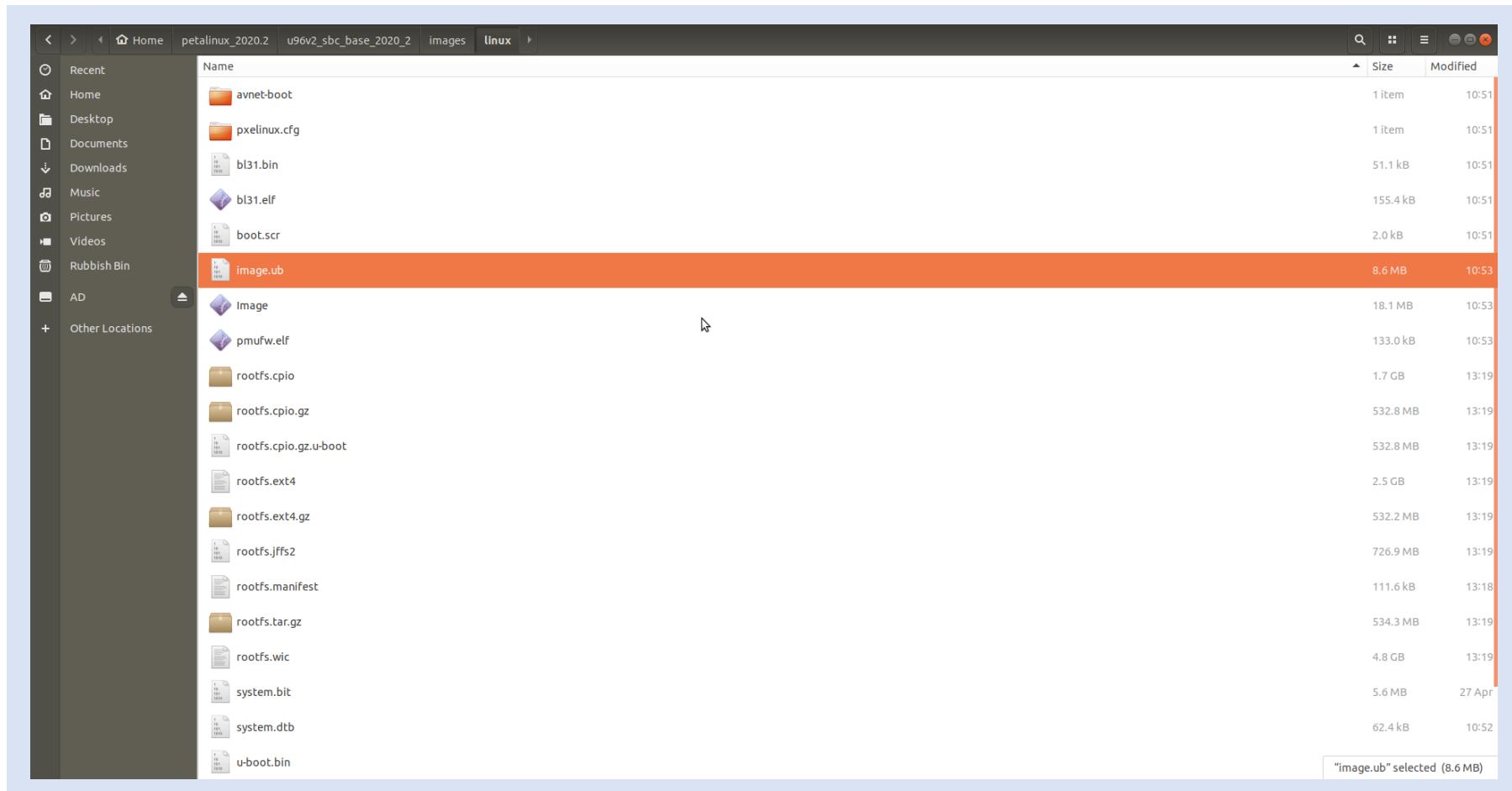


A terminal window showing the output of a `petalinux-build -c avnet-image-full` command. The output indicates the build process has completed successfully, with all tasks run and images copied to the tftpboot directory.

```
File Edit View Search Terminal Help  
adiuvo@Adiuvo:~/petalinux_2020.2/u96v2_sbc_base_2020_2$ petalinux-build -c avnet-image-full  
INFO: Sourcing build tools  
[INFO] Building avnet-image-full  
[INFO] Sourcing build environment  
[INFO] Generating workspace directory  
INFO: bitbake avnet-image-full  
Loading cache: 100% [#####################################################################] Time: 0:00:00  
Loaded 4294 entries from dependency cache  
Parsing recipes: 100% [#####################################################################] Time: 0:00:04  
Parsing of 3027 .bb files complete (3023 Cached, 4 parsed). 4297 targets, 176 skipped, 0 masked, 0 errors.  
NOTE: Resolving any missing task queue dependencies  
Initialising tasks: 100% [#####################################################################] Time: 0:00:13  
Checking sstate mirror object availability: 100% [#####################################################################] Time: 0:00:16  
Sstate summary: Wanted 2932 Found 2144 Missed 788 Current 876 (73% match, 79% complete)  
NOTE: Executing Tasks  
NOTE: Setscene tasks completed  
NOTE: Tasks Summary: Attempted 11639 tasks of which 9194 didn't need to be rerun and all succeeded.  
INFO: Failed to copy built images to tftp dir: /tftpboot  
[INFO] Successfully built avnet-image-full  
adiuvo@Adiuvo:~/petalinux_2020.2/u96v2_sbc_base_2020_2$
```

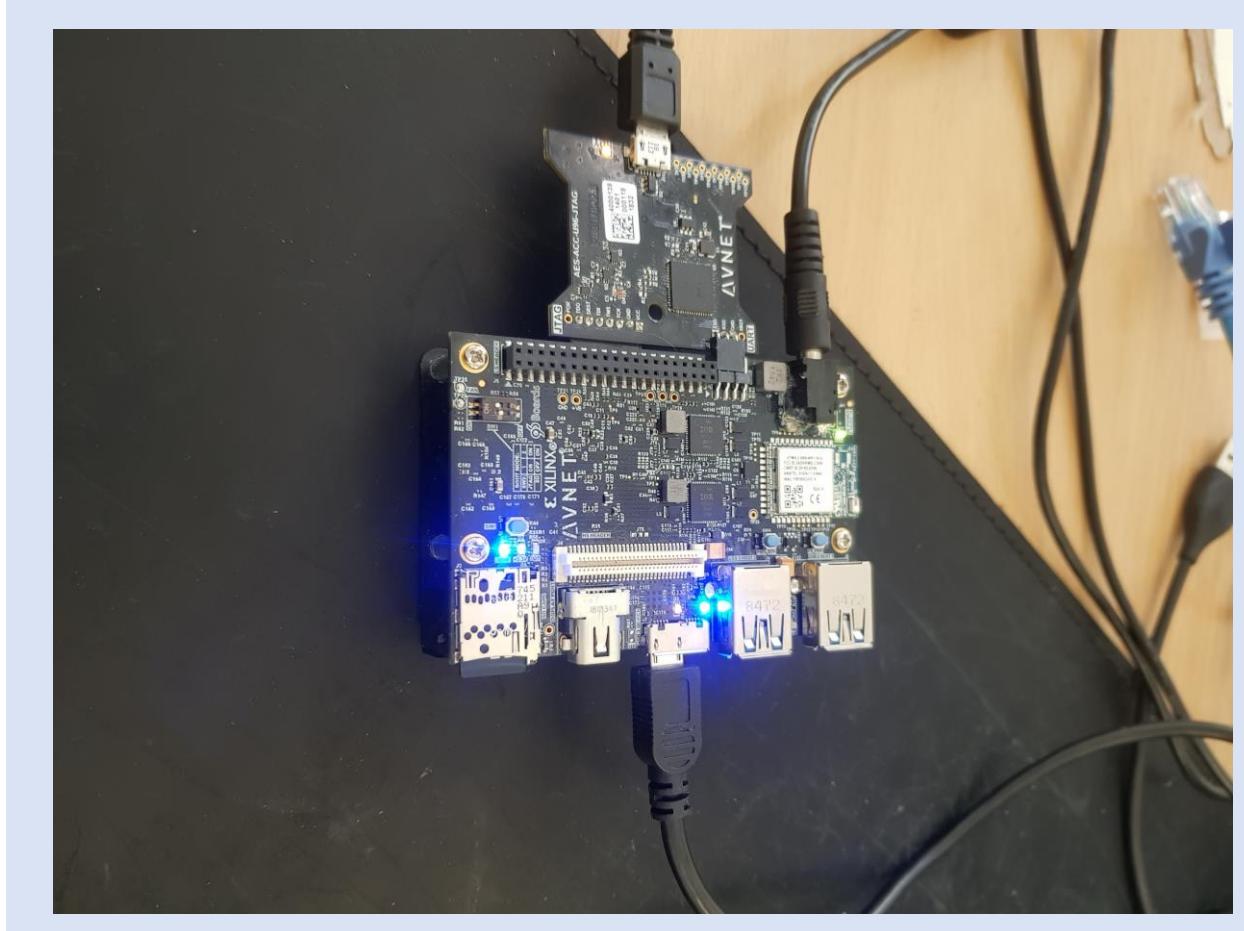
Lab 3: Creating Embedded Linux Solutions

Step 19 – From the <project>/images/linux directory copy image.ub to the SD Card boot partition



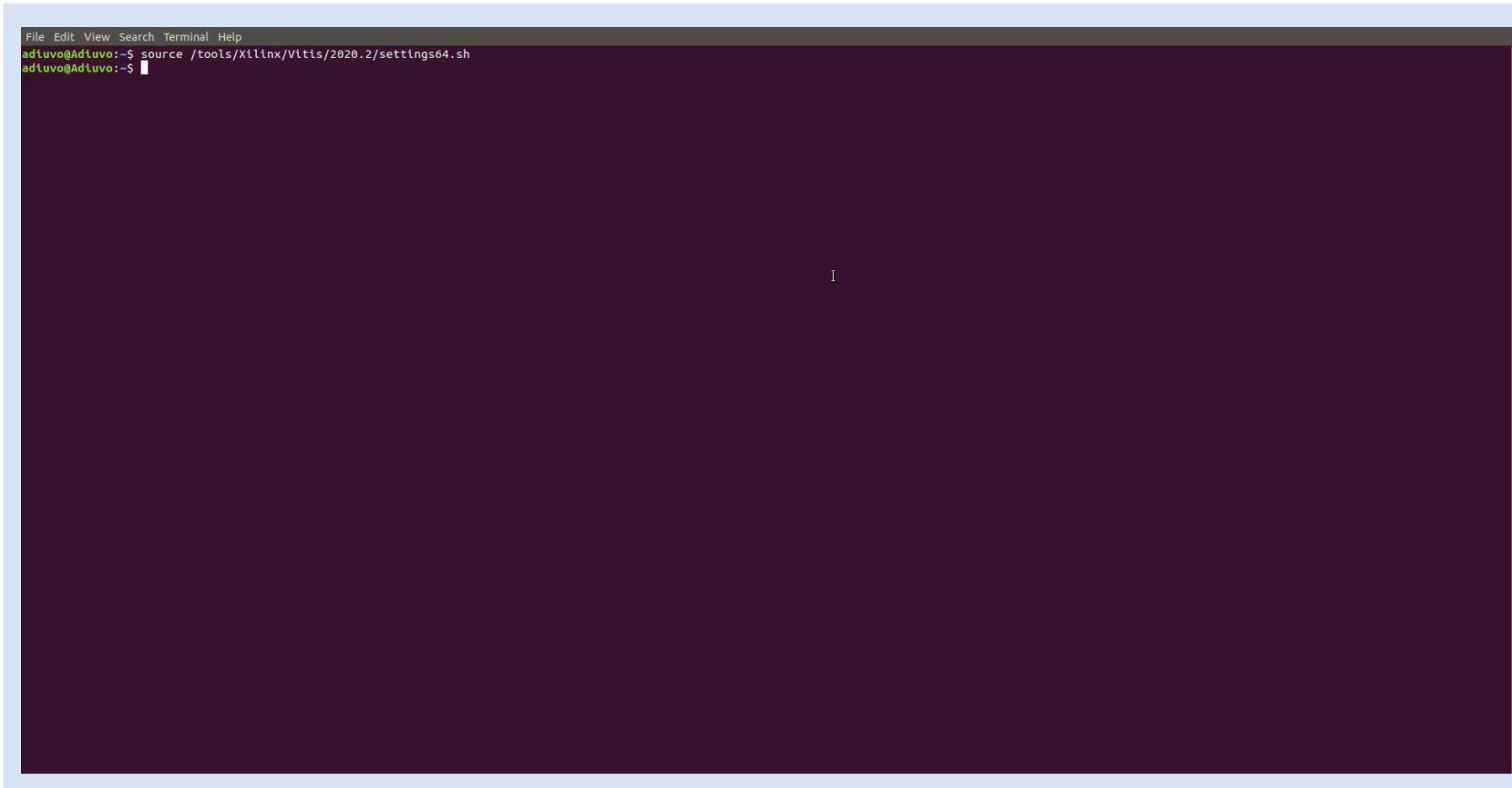
Lab 3: Creating Embedded Linux Solutions

Step 20 – Insert the SD Card into the Ultra96V2 and connect the JTAG pod, connect a USB from the USB connector on the front to your development machine and power on



Lab 3: Creating Embedded Linux Solutions

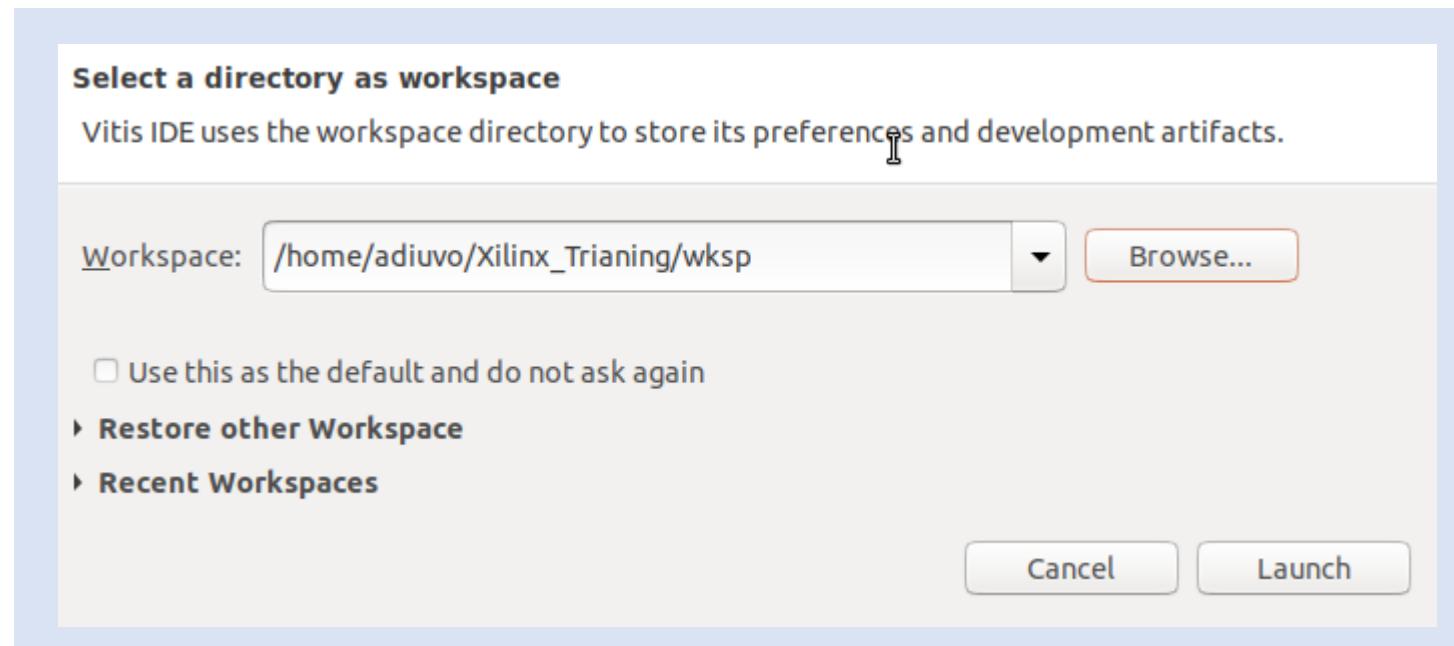
Step 21 – On the linux development machine / virtual machine. Open a terminal and type
Source `/opt/tools/Xilinx/Vitis/2020.2/settings64.sh` This assumes installation path is `/opt/`



```
File Edit View Search Terminal Help
adiuvo@Adieuvo:~$ source /opt/tools/Xilinx/Vitis/2020.2/settings64.sh
adiuvo@Adieuvo:~$ █
```

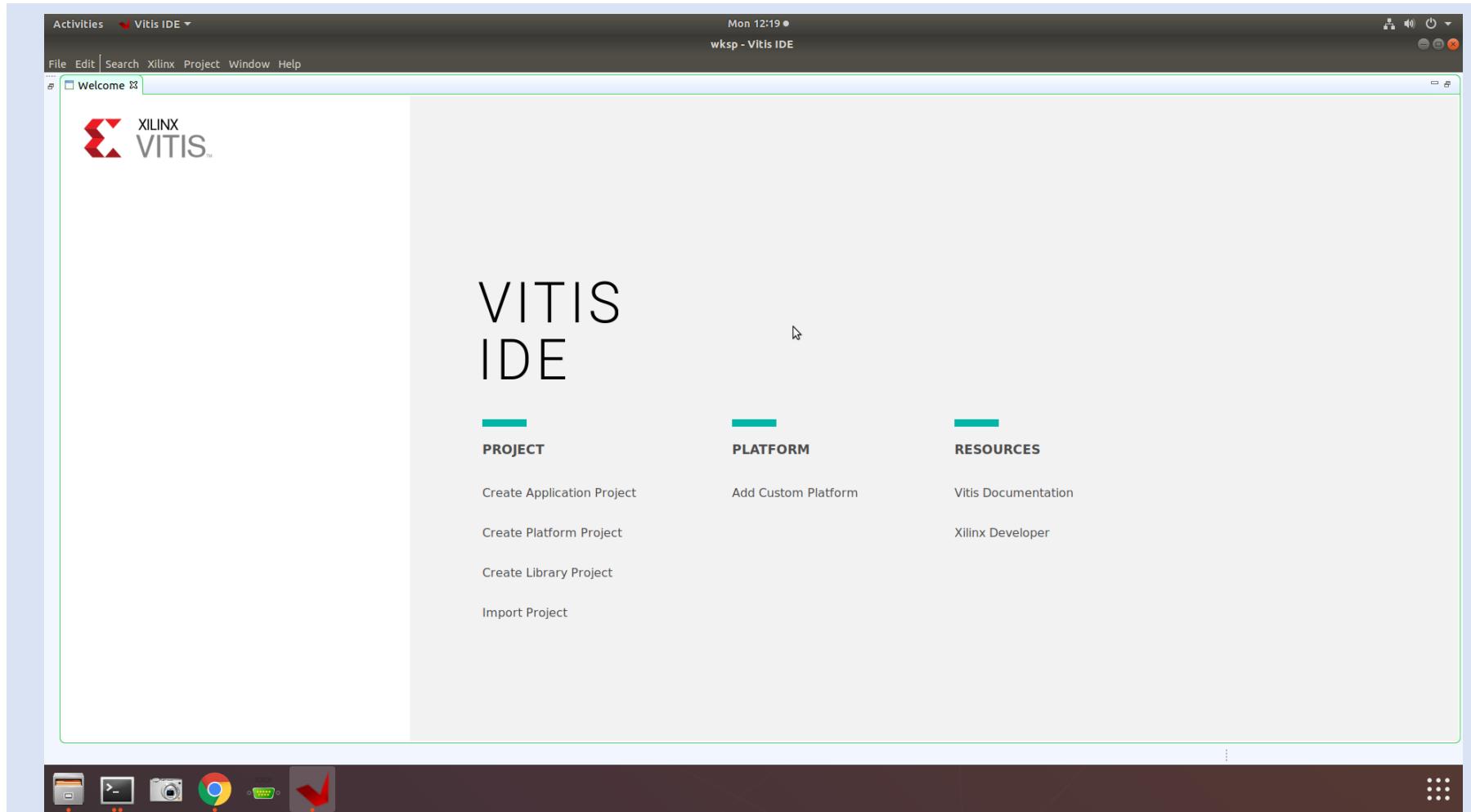
Lab 3: Creating Embedded Linux Solutions

Step 22 – Vitis will start and ask for the workspace, create a new directory using browse to store the projects.



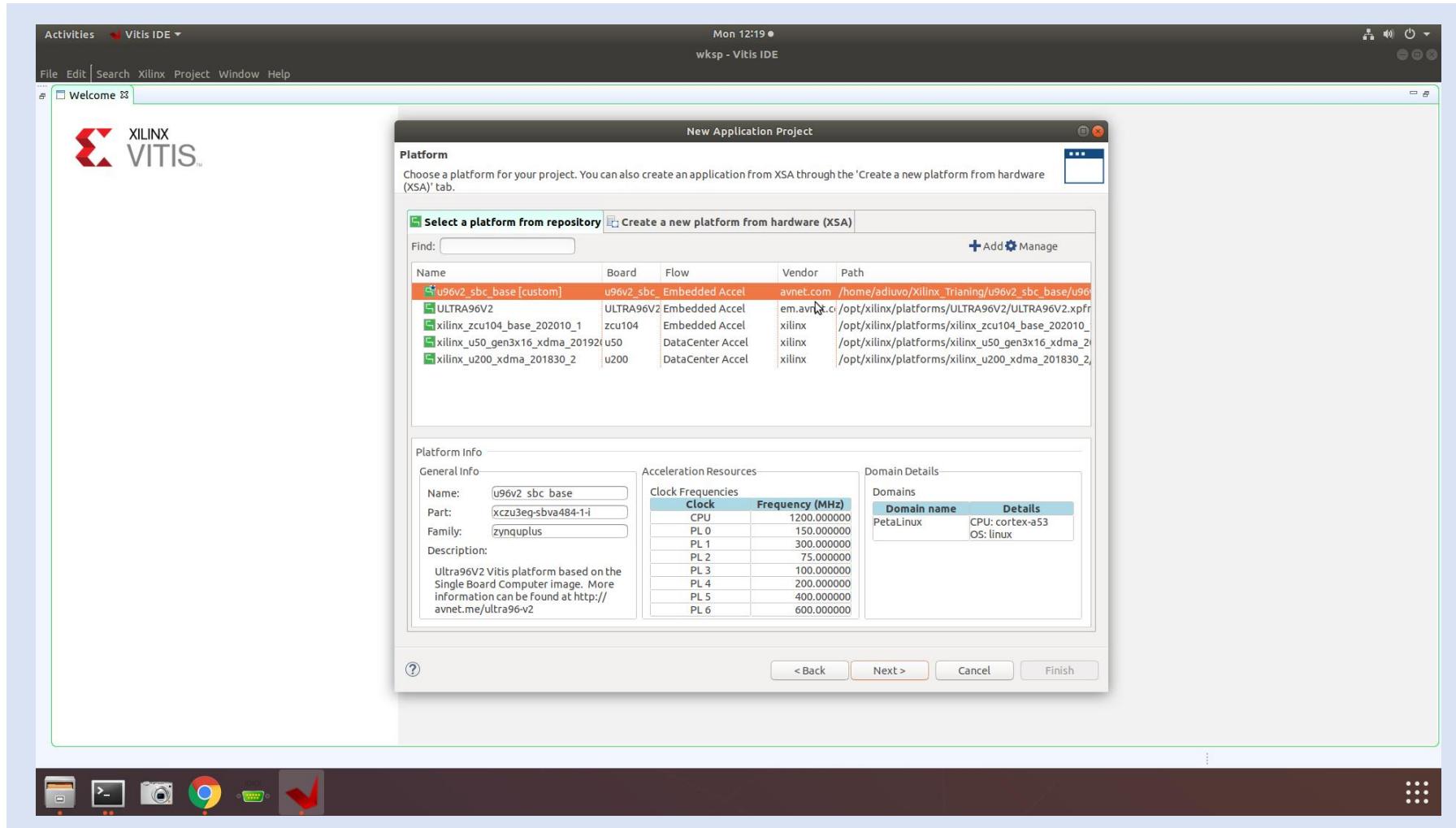
Lab 3: Creating Embedded Linux Solutions

Step 23 – Select Create Application Project



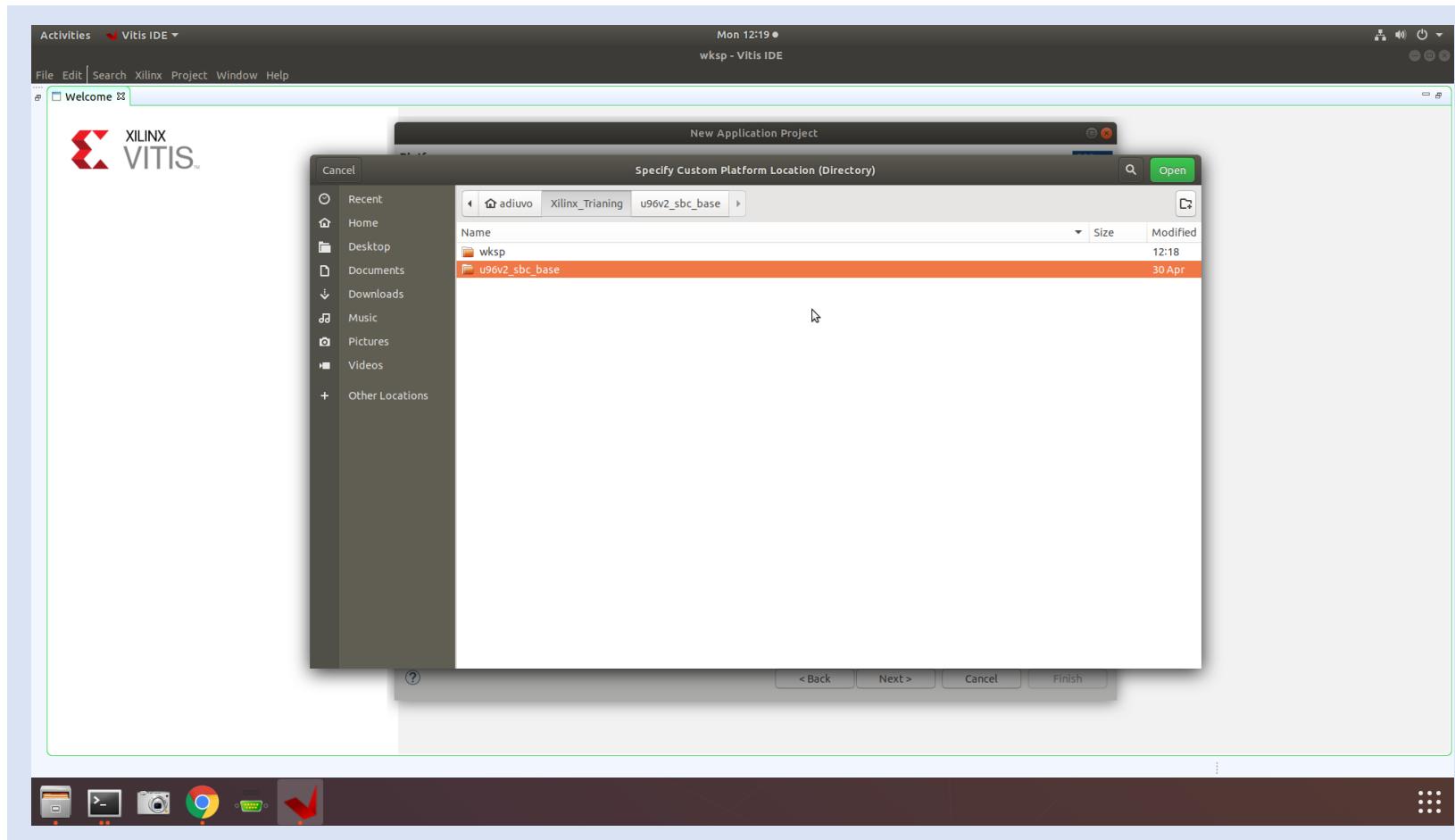
Lab 3: Creating Embedded Linux Solutions

Step 24 – On the Platform page select + Add and point it to the extracted U96 we downloaded earlier



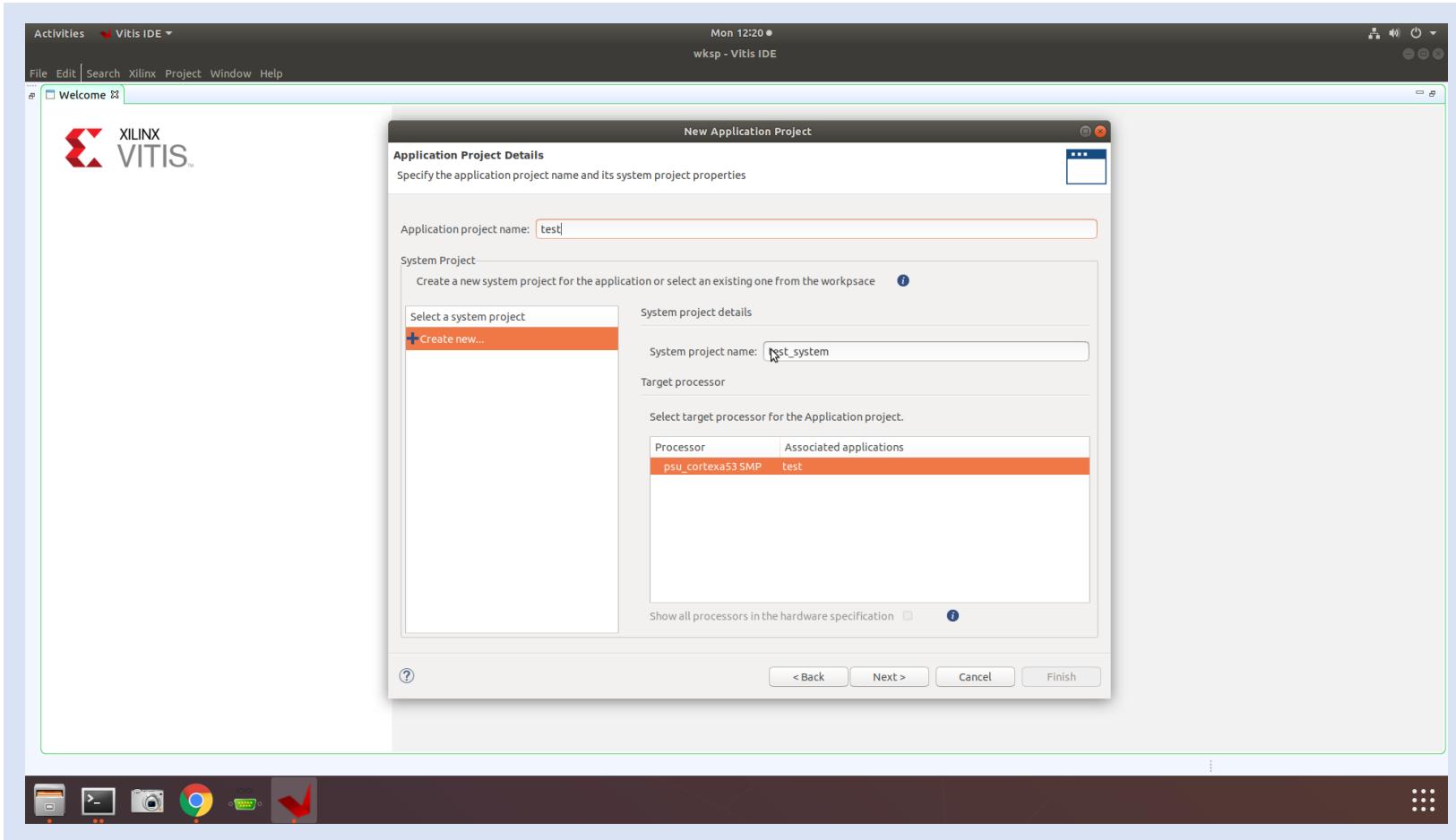
Lab 3: Creating Embedded Linux Solutions

Step 25 – Select the Top-Level directory of the Ultra96V2 Platform – Select that for use click next



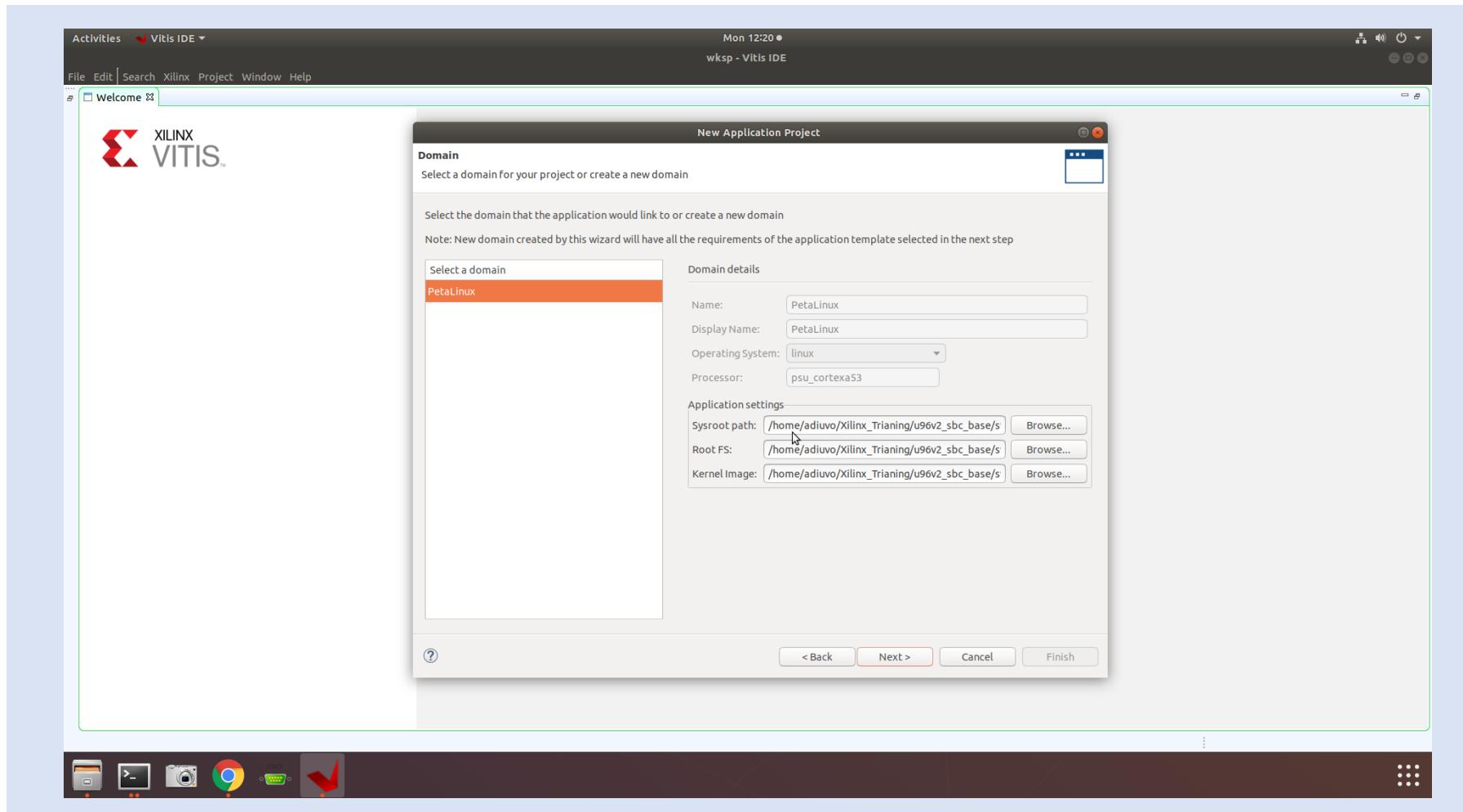
Lab 3: Creating Embedded Linux Solutions

Step 26 – Enter a name for the project



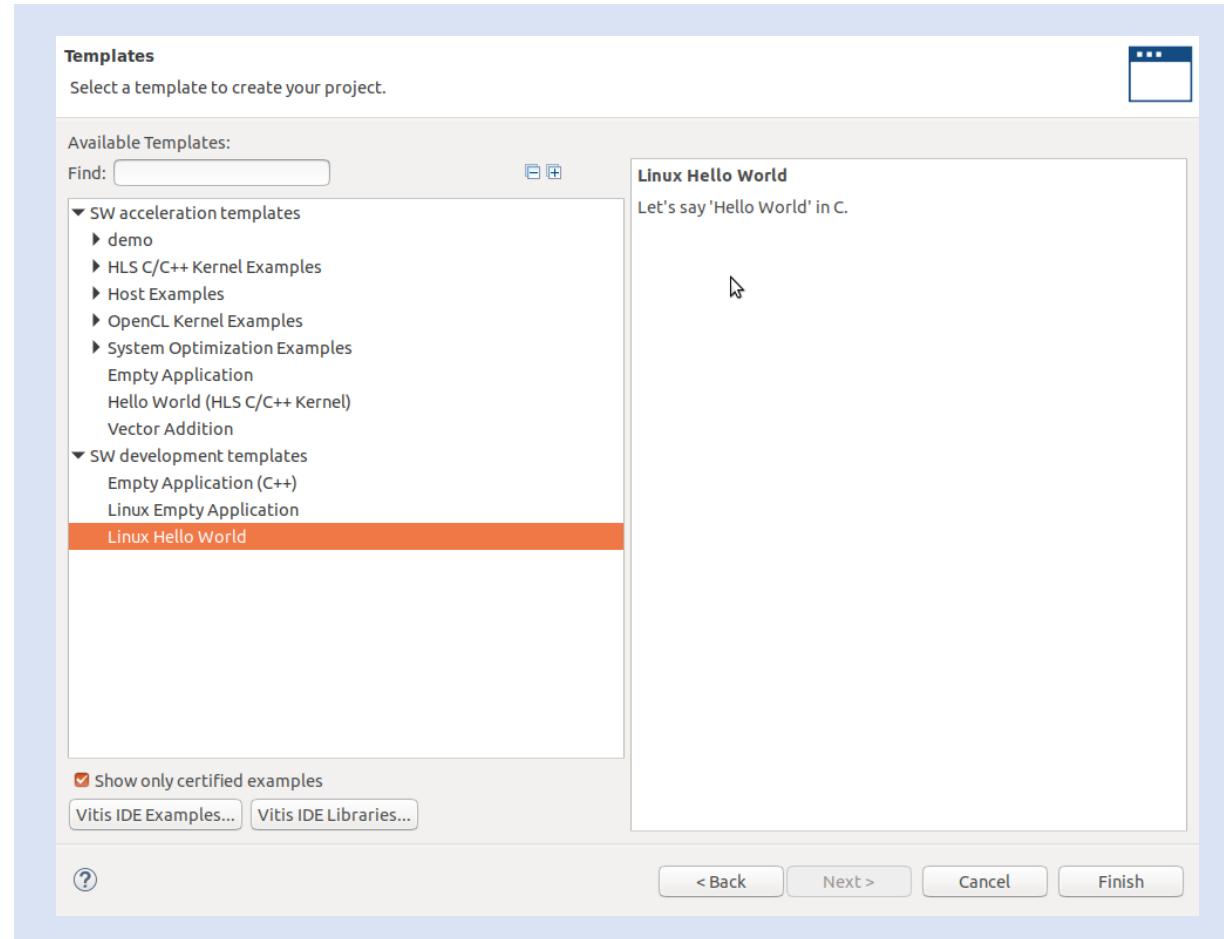
Lab 3: Creating Embedded Linux Solutions

Step 27 – Click Next on the Domain page



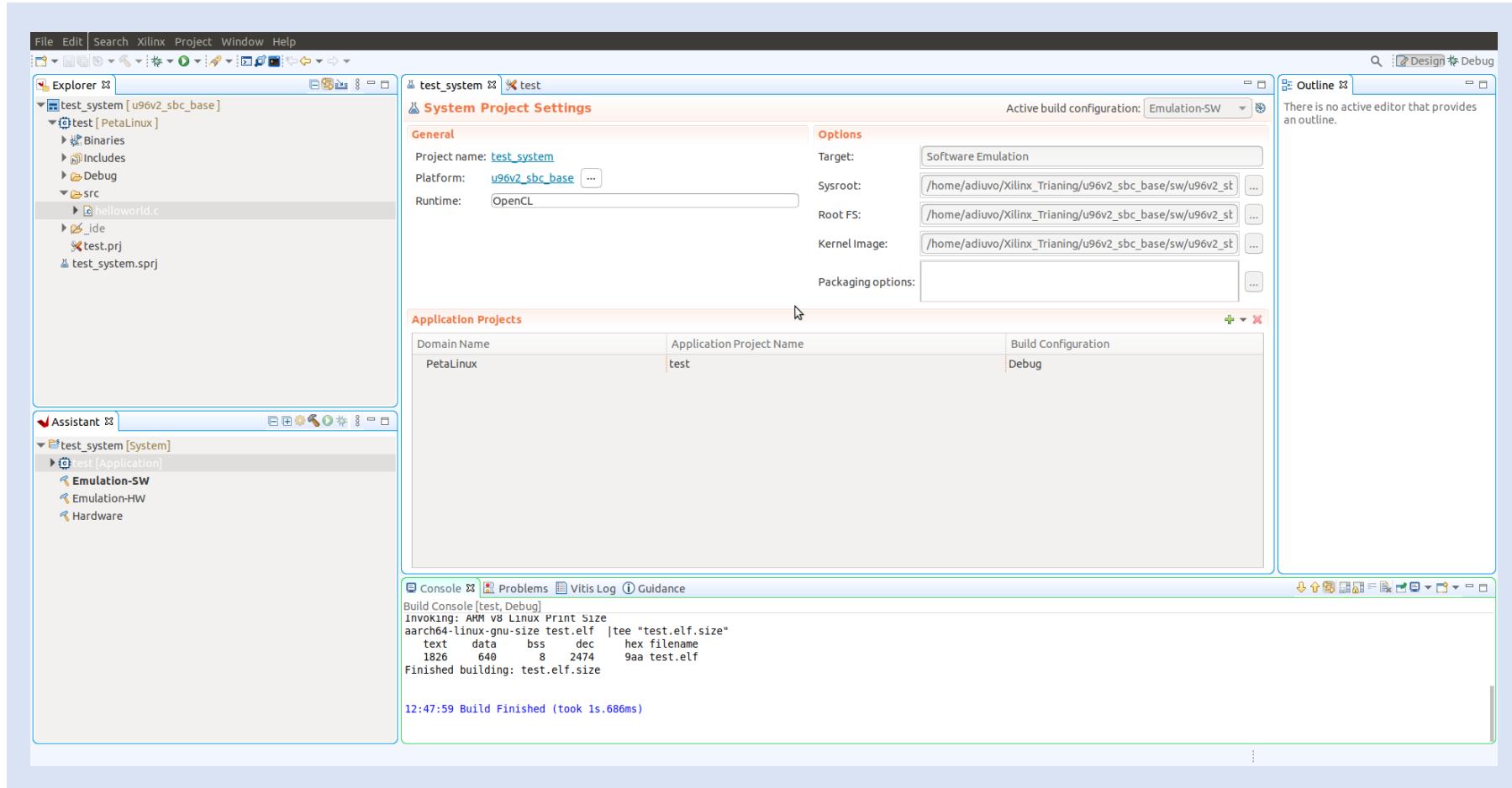
Lab 3: Creating Embedded Linux Solutions

Step 28 – Select the Linux Hello World application and click finish



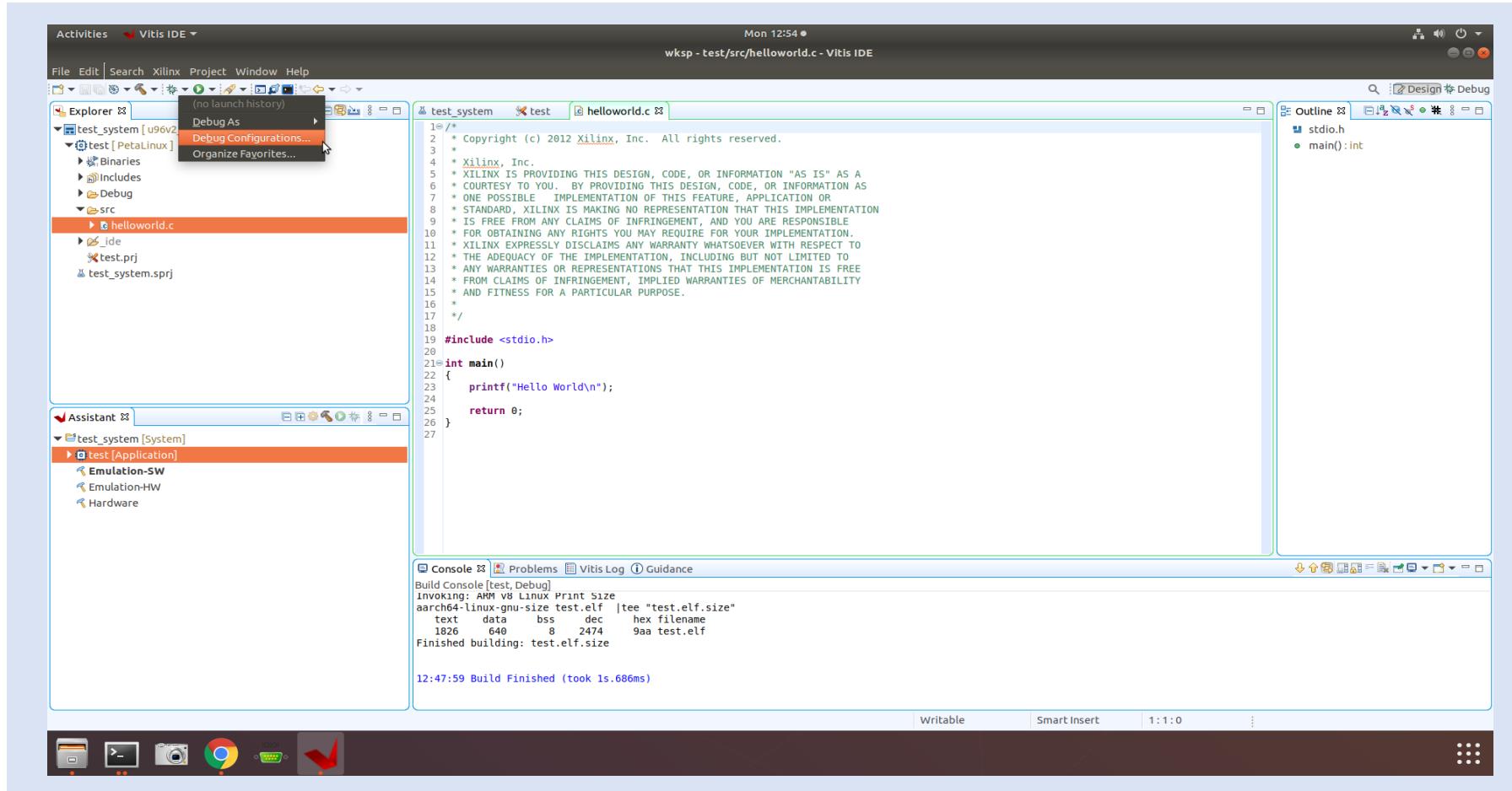
Lab 3: Creating Embedded Linux Solutions

Step 29 – Click on the Hammer to build the application



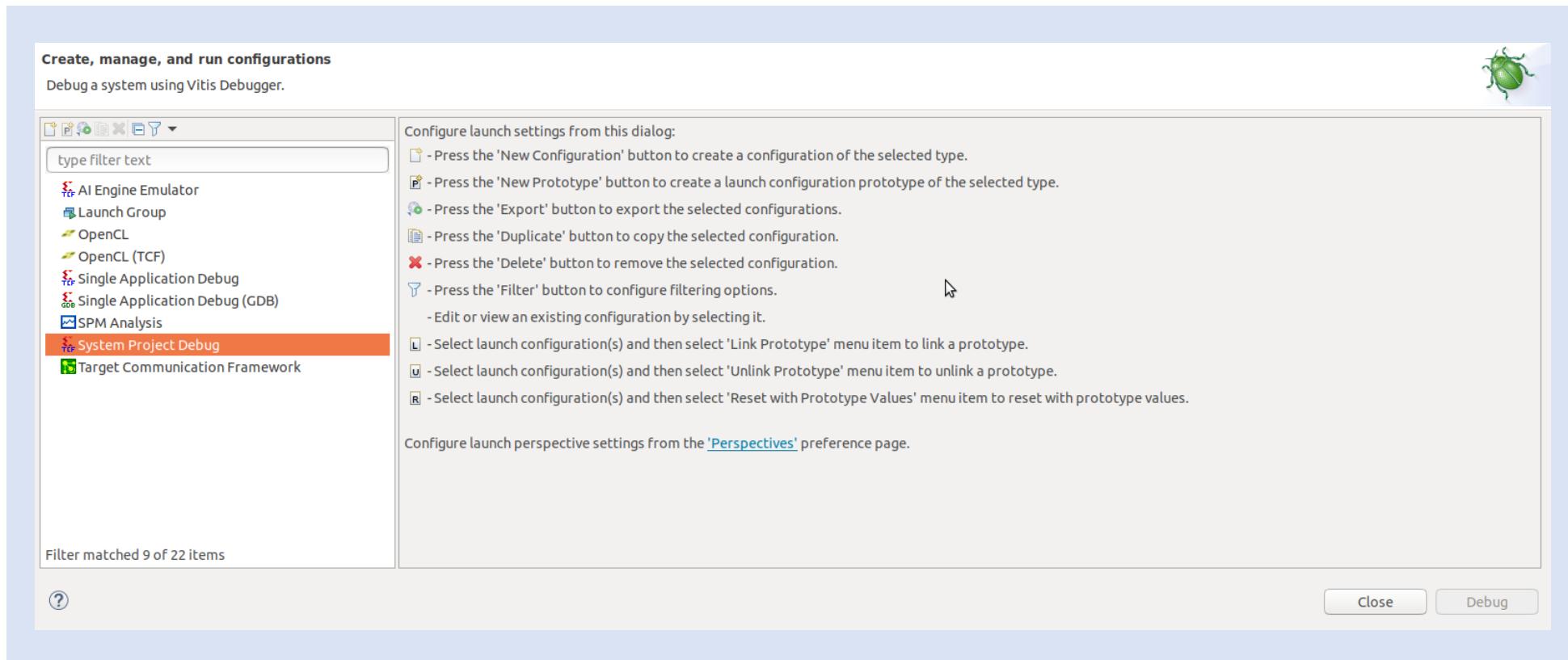
Lab 3: Creating Embedded Linux Solutions

Step 30 – Select the arrow by the green debug icon and select debug configuration



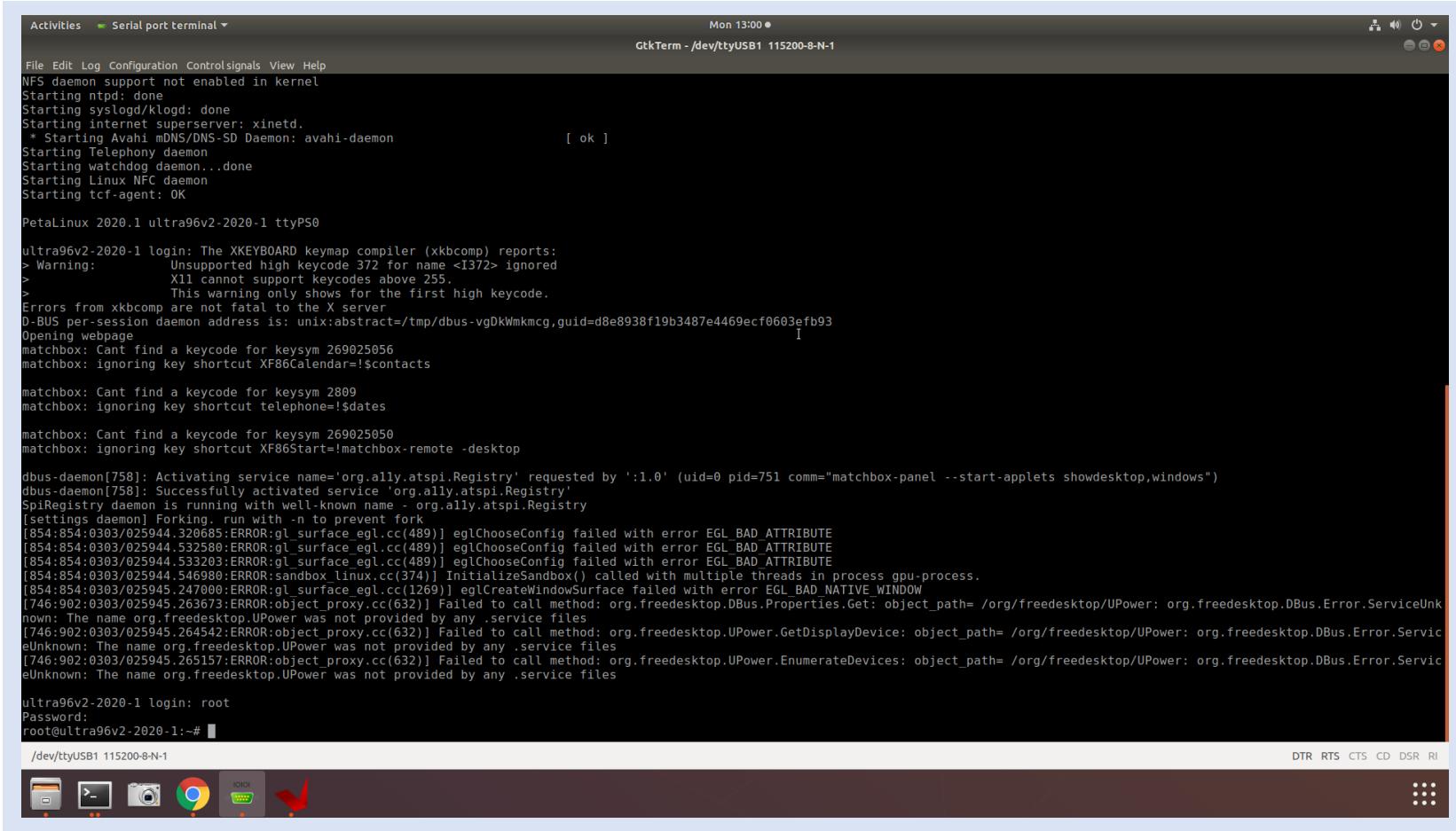
Lab 3: Creating Embedded Linux Solutions

Step 31 – Select System Project Debug and click new



Lab 3: Creating Embedded Linux Solutions

Step 32 – In the Terminal window (115200,1,N,8) log in password and username are root



The screenshot shows a terminal window titled "Serial port terminal" running on a PetaLinux 2020.1 system. The window displays a log of system boot-up, including the starting of various daemons like ntpd, klogd, and avahi-daemon. It also shows the configuration of a keyboard keymap compiler (xkbcomp) which ignores unsupported high keycodes above 255. The log continues with errors from xkbcomp and dbus-daemon, and ends with a successful root login.

```
Activities Serial port terminal • Mon 13:00 • GtkTerm - /dev/ttyUSB1 115200-8-N-1

File Edit Log Configuration Controlsignals View Help
NFS daemon support not enabled in kernel
Starting ntpd: done
Starting syslogd/klogd: done
Starting internet superserver: xinetd.
 * Starting Avahi mDNS/DNS-SD Daemon: avahi-daemon [ ok ]
Starting Telephony daemon
Starting watchdog daemon...done
Starting Linux NFC daemon
Starting tcf-agent: OK

PetaLinux 2020.1 ultra96v2-2020-1 ttyPS0

ultra96v2-2020-1 login: The XKEYBOARD keymap compiler (xkbcomp) reports:
> Warning:      Unsupported high keycode 372 for name <I372> ignored
>           X11 cannot support keycodes above 255.
>           This warning only shows for the first high keycode.
Errors from xkbcomp are not fatal to the X server
D-BUS per-session daemon address is: unix:abstract=/tmp/dbus-vgDkWmkmcg,guid=d8e8938f19b3487e4469ecf0603efb93
Opening webpage
matchbox: Cant find a keycode for keysym 269025056
matchbox: ignoring key shortcut XF86Calendar=!$contacts

matchbox: Cant find a keycode for keysym 2809
matchbox: ignoring key shortcut telephone=!$dates

matchbox: Cant find a keycode for keysym 269025050
matchbox: ignoring key shortcut XF86Start=!matchbox-remote -desktop

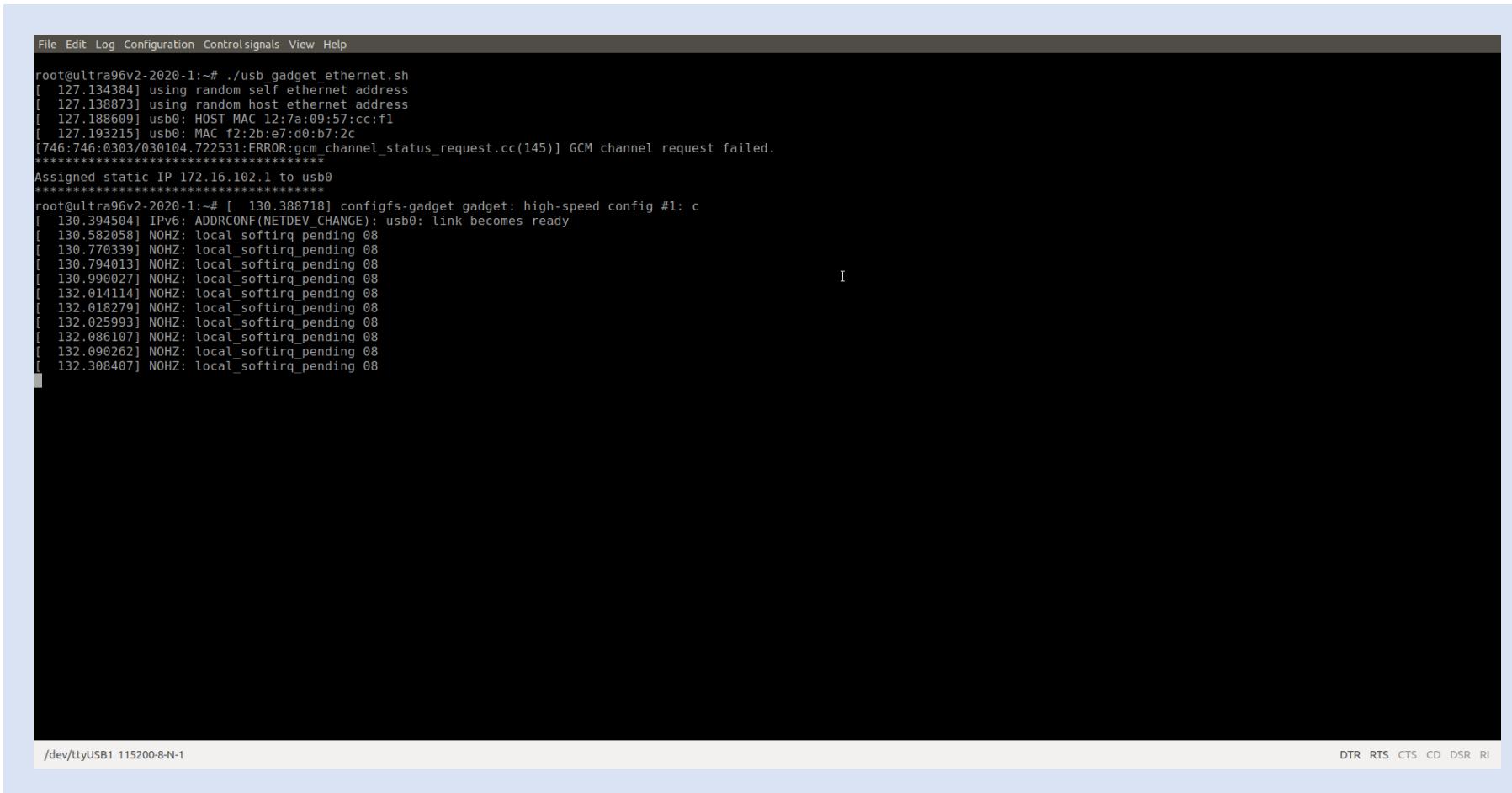
dbus-daemon[758]: Activating service name='org.ally.atspi.Registry' requested by ':1.0' (uid=0 pid=751 comm="matchbox-panel --start-applets showdesktop,windows")
dbus-daemon[758]: Successfully activated service 'org.ally.atspi.Registry'
SpiRegistry daemon is running with well-known name org.ally.atspi.Registry
[settings daemon] Forking. run with -n to prevent fork
[854:854:0303/025944.320685:ERROR:gl_surface_e gl.cc(489)] eglChooseConfig failed with error EGL_BAD_ATTRIBUTE
[854:854:0303/025944.532580:ERROR:gl_surface_e gl.cc(489)] eglChooseConfig failed with error EGL_BAD_ATTRIBUTE
[854:854:0303/025944.533203:ERROR:gl_surface_e gl.cc(489)] eglChooseConfig failed with error EGL_BAD_ATTRIBUTE
[854:854:0303/025944.546980:ERROR:sandbox_linux.cc(374)] InitializeSandbox() called with multiple threads in process gpu-process.
[854:854:0303/025945.247900:ERROR:gl_surface_e gl.cc(1269)] eglCreateWindowSurface failed with error EGL_BAD_NATIVE_WINDOW
[746:902:0303/025945.263673:ERROR:object_proxy.cc(632)] Failed to call method: org.freedesktop.UPower.Properties.Get: object_path= /org/freedesktop/UPower: org.freedesktop.DBus.Error.ServiceUnknown: The name org.freedesktop.UPower was not provided by any .service files
[746:902:0303/025945.264542:ERROR:object_proxy.cc(632)] Failed to call method: org.freedesktop.UPower.GetDisplayDevice: object_path= /org/freedesktop/UPower: org.freedesktop.DBus.Error.ServiceUnknown: The name org.freedesktop.UPower was not provided by any .service files
[746:902:0303/025945.265157:ERROR:object_proxy.cc(632)] Failed to call method: org.freedesktop.UPower.EnumerateDevices: object_path= /org/freedesktop/UPower: org.freedesktop.DBus.Error.ServiceUnknown: The name org.freedesktop.UPower was not provided by any .service files

ultra96v2-2020-1 login: root
Password:
root@ultra96v2-2020-1:~#
```

At the bottom of the terminal window, there is a status bar showing serial port settings: DTR, RTS, CTS, CD, DSR, RI, and a path indicator: /dev/ttyUSB1 115200-8-N-1. Below the status bar is a toolbar with several icons, including a terminal, a file manager, a browser, and system monitoring tools.

Lab 3: Creating Embedded Linux Solutions

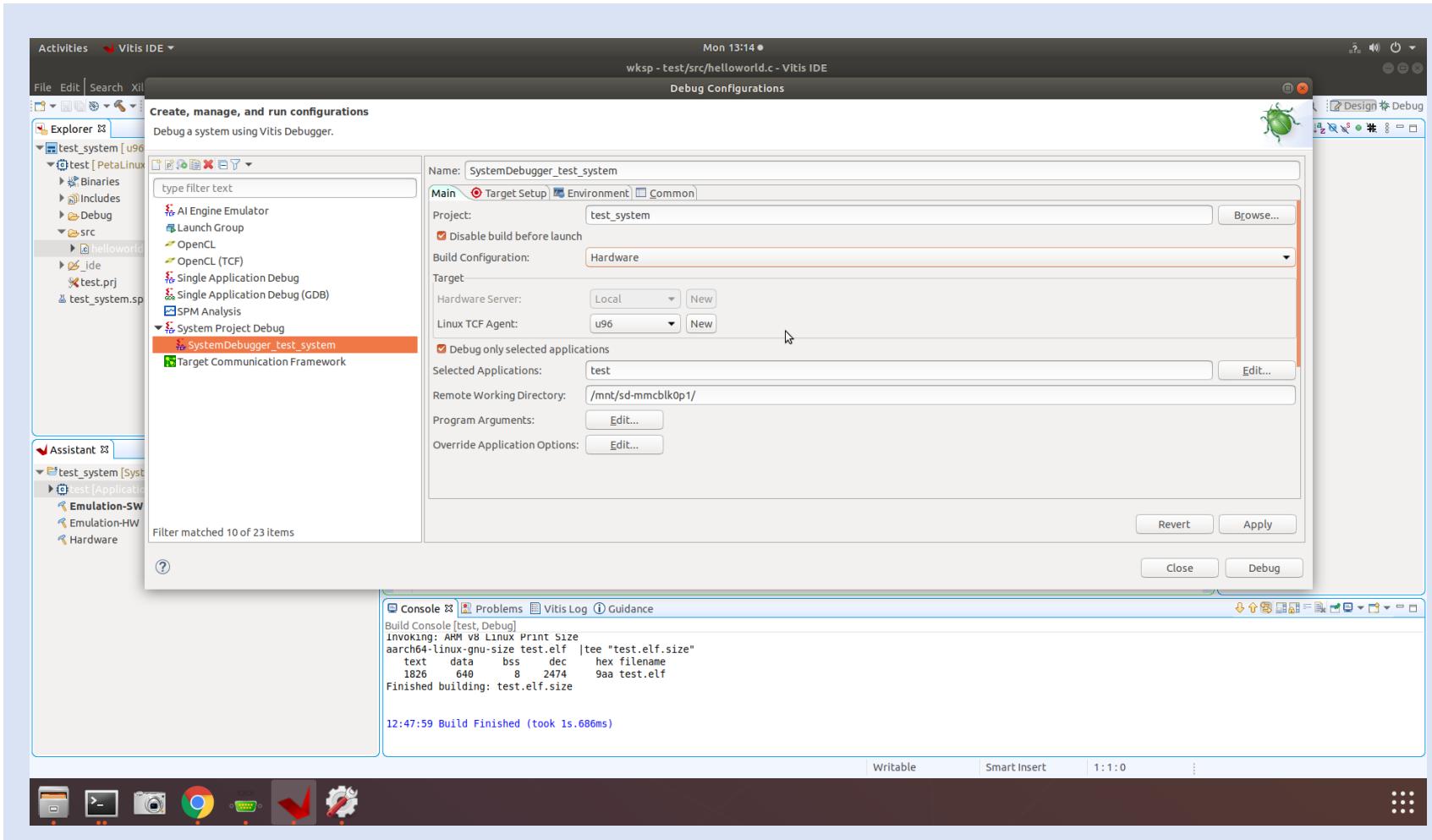
Step 33 – Enter the command `./usb_gadget_ethernet.sh` you may need to use VI to edit IP address to align with your network. By default, it is 192.168.3.1



```
File Edit Log Configuration Controlsignals View Help
root@ultra96v2-2020-1:~# ./usb_gadget_ethernet.sh
[ 127.134384] using random self ethernet address
[ 127.138873] using random host ethernet address
[ 127.188609] usb0: HOST MAC 12:7a:09:57:cc:f1
[ 127.193215] usb0: MAC f2:2b:e7:d0:b7:2c
[746:746:0303/030104.722531:ERROR:gcm_channel_status_request.cc(145)] GCM channel request failed.
*****
Assigned static IP 172.16.102.1 to usb0
*****
root@ultra96v2-2020-1:~# [ 130.388718] configfs-gadget gadget: high-speed config #1: c
[ 130.394504] IPv6: ADDRCONF(NETDEV_CHANGE): usb0: link becomes ready
[ 130.582058] NOHZ: local_softirq_pending 08
[ 130.770339] NOHZ: local_softirq_pending 08
[ 130.794013] NOHZ: local_softirq_pending 08
[ 130.990027] NOHZ: local_softirq_pending 08
[ 132.014114] NOHZ: local_softirq_pending 08
[ 132.018279] NOHZ: local_softirq_pending 08
[ 132.025993] NOHZ: local_softirq_pending 08
[ 132.086107] NOHZ: local_softirq_pending 08
[ 132.090262] NOHZ: local_softirq_pending 08
[ 132.308407] NOHZ: local_softirq_pending 08
I
/dev/ttyUSB1 115200-8-N-1
DTR RTS CTS CD DSR RI
```

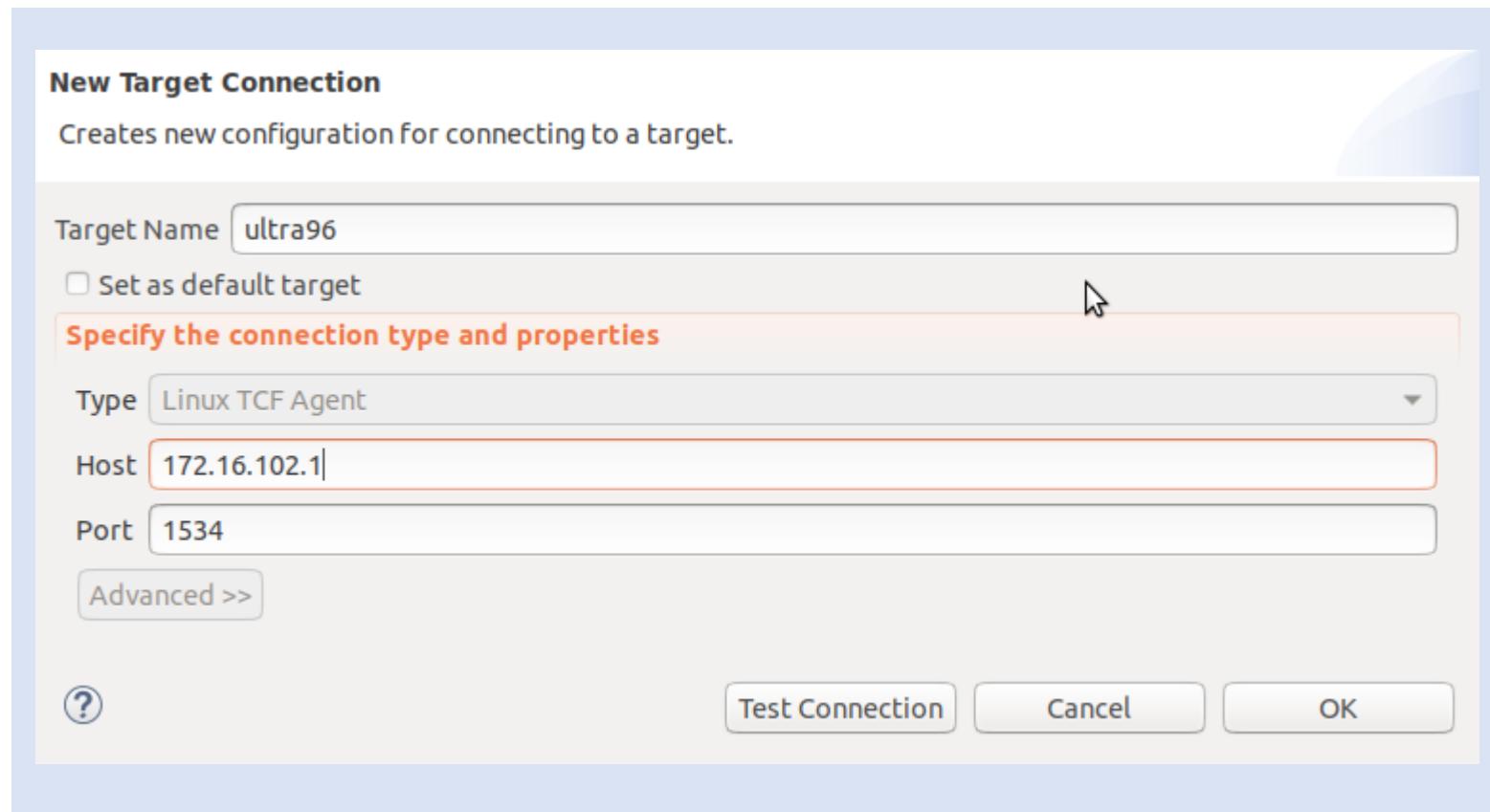
Lab 3: Creating Embedded Linux Solutions

Step 34 – Change the build configuration to Hardware and for Linux TCF Agent select new



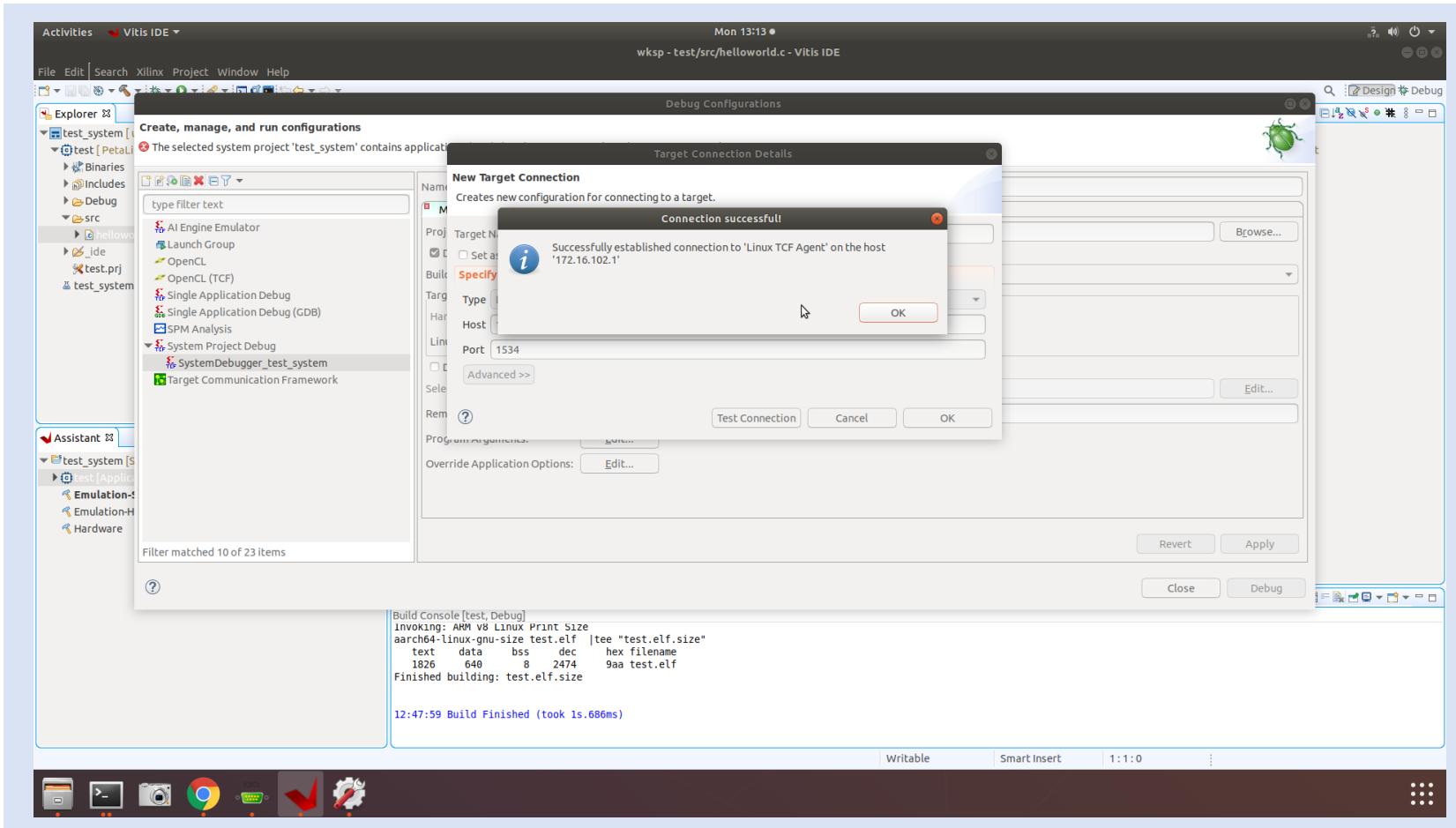
Lab 3: Creating Embedded Linux Solutions

Step 35 – Enter a name, and enter the USB Gadget Ethernet IP address – Click test connection to verify the board can be seen.



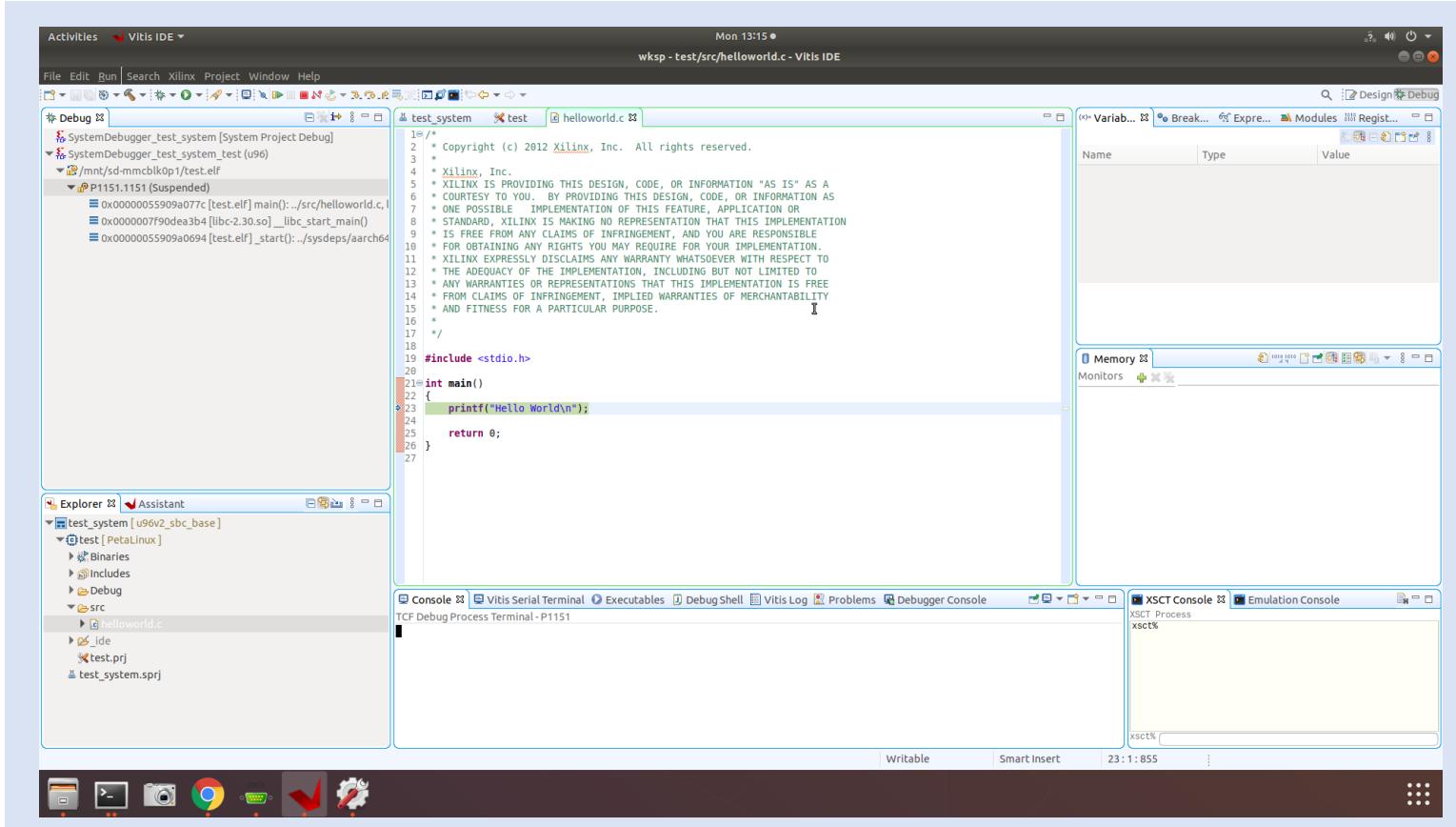
Lab 3: Creating Embedded Linux Solutions

Step 36 – Click OK when the connection is successful, click OK on the target connection and the click debug



Lab 3: Creating Embedded Linux Solutions

Step 37 – The application will be downloaded to the target and paused for execution, press the run button



Lab 3: Creating Embedded Linux Solutions

Step 38 – Click on the run button and you will see the hello world appear in the console.

