KRIA board Ubuntu LTS 22.04 Install

Vincent Conus*

2023-12-22

Contents

0.1 Preparing and booting a Ubuntu 22.04 media

An official Ubuntu image exists and is provided by Xilinx, allowing the OS installation to be quick and straightforward. Ubuntu is a common and easy to use distribution. Furthermore, it allows to install ROS2 as a package, which is most convenient and will be done later in this guide.

Once the image has been downloaded at Canonical's page we can flash it onto the SD card, with the following instructions.

> DANGER: The next part involve the `dd` command writing on disks!!! As always with the `dd` command, thou have to be VERY careful on what arguments thou give. Selecting the wrong disk will result on the destruction of thy data !! If you are unsure of what to do, seek assistance !

With the image available on thy machine and a SD card visible as `/dev/sda` device[1] one can simply run the `dd` command as follow to write the image to a previously formatted drive (here `/dev/sda`):

```
unxz iot-limerick-kria-classic-desktop-2204-x07-20230302-63.img.xz
sudo dd if=iot-limerick-kria-classic-desktop-2204-x07-20230302-63.img \
     of=/dev/sda status=progress bs=8M && sync
```

Once the SD card is flashed and put back in the board, the micro-USB cable can be connected from the PC to the board. It is then possible to connect to the board in serial with an appropriate tool, for example `picocom`, as in the following example (the serial port that "appeared" was the `/dev/ttyUSB1` in this case, and the 115200 bit-rate is the default value for the board):

*vincent.conus@protonmail.com

[1] Again, it is critical to be 100\the correct device!

```
sudo picocom /dev/ttyUSB1 -b 115200
```

In my case, I am using Emacs's `serial-term`:

```
M-x serial-term RET /dev/ttyUSB1 RET 115200 RET
```

The default username / password pair for the very first boot is `ubuntu` and `ubuntu`. You will then be prompted to enter a new passwork.

Once logged in, it is typically easier and more convenient to connect the board using SSH. When the board is connected to the network, it is possible to know it's IP address with the `IP` command; then it is possible to connect to the board with ssh, as follow (example, with the first command to be run on the board and the second one on the host PC, both without the first placeholder hostnames):

```
kria# ip addr

host# ssh ubuntu@192.168.4.11
```

0.2 Network and admin setups

This section presents a variety of extra convenience configurations that can be used when setting-up the Kria board.

0.2.1 Static IP address

A static IP can be set by writing the following configuration into your `netplan` configuration file[2].

The name of the files might vary:

```
sudo chmod 0600 /etc/netplan/50-cloud-init.yaml
sudo nano /etc/netplan/50-cloud-init.yaml
```

You can then set the wanted IP as follow[3]:

```
network:
  renderer: NetworkManager
  version: 2
  ethernets:
    eth0:
      dhcp4: false
      addresses:
        - 192.168.11.107/24
      routes:
        - to: default
          via: 192.168.11.1
      nameservers:
        addresses: [192.168.11.1]
```

Finally, the change in settings can be applied as follow:

```
sudo netplan apply
```

[2] The `chmod` command is used to update the permissions and silence some warnings

[3] For the routing part, it is key to have the `to` with a `'-'` in front of it; and then the `via` without, but alligned with the `t`.

0.2.2 Proxy and DNS

An issue that can occur when connecting the board to the internet is the conflicting situation with the university proxy. Indeed, as the network at Nanzan University requires to go through a proxy, some DNS errors appeared.

In that case, it might become needed to setup the proxy for the school.

This can be done as follow, by exporting a https base proxy configuration containing you AXIA credentials (this is specific to Nanzan University IT system), then by consolidating the configuration for other types of connections in the `bashrc`:

```
export https_proxy="http://<AXIA_username>:\
       <AXIA_psw>@proxy.ic.nanzan-u.ac.jp:8080"

echo "export http_proxy=\""$https_proxy"\"" >> ~/.bashrc
echo "export https_proxy=\""$https_proxy"\"" >> ~/.bashrc
echo "export ftp_proxy=\""$https_proxy"\"" >> ~/.bashrc
echo "export no_proxy=\"localhost, 127.0.0.1,::1\"" >> ~/.bashrc
```

Eventually the board can be rebooted in order for the setup to get applied cleanly.

0.2.3 `root` password

WARNING: Depending on your use-case, the setup presented in this subsection can be a critical security breach as it remove the need for a root password to access the admin functions of the board's Linux. When in doubt, do not apply this configuration!!

If you board does not hold important data and is available to you only, for test or development, it might be convenient for the `sudo` tool to not ask for the password all the time. This change can be done by editing the sudoers file, and adding the parameter `NOPASSWD` at the `sudo` line:

```
sudo visudo

%sudo   ALL=(ALL:ALL) NOPASSWD: ALL
```

Again, this is merely a convenience setup for devices staying at you desk. If the board is meant to be used in any kind of production setup, a password should be set for making administration tasks.

With all of these settings, you should be able to update the software of your board without any issues:

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo reboot now
```

0.2.4 Purging `snap`

As the desktop-specific software are not used at all in the case of our project, there are some packages that can be purges in order for the system to become more lightweight.

In particular, the main issue with Ubuntu systems is the forced integration of Snap packages. Here are the command to use in order to remove all of that. These steps take a lot of time and need to be executed in that specific order[4], but the system fan runs sensibly slower without all of this stuff:

[4] The `snap` packages depends on each others. Dependencies cannot be remove before the package(s) that depends on them, thus the specific delete order.

```
sudo systemctl disable snapd.service
sudo systemctl disable snapd.socket
sudo systemctl disable snapd.seeded.service

sudo snap list #show installed package, remove then all:
sudo snap remove --purge firefox
sudo snap remove --purge gnome-3-38-2004
sudo snap remove --purge gnome-42-2204
sudo snap remove --purge gtk-common-themes
sudo snap remove --purge snapd-desktop-integration
sudo snap remove --purge snap-store
sudo snap remove --purge bare
sudo snap remove --purge core20
sudo snap remove --purge core22
sudo snap remove --purge snapd
sudo snap list # check that everything is uninstalled

sudo rm -rf /var/cache/snapd/
sudo rm -rf ~/snap
sudo apt autoremove --purge snapd

# check once more that there is no more snap on the system
systemctl list-units | grep snapd
```

0.2.5 Other unused heavy packages

Some other pieces of software can safely be removed since the desktop is not to be used:

```
sudo apt-get autoremove --purge yaru-theme-icon \
     fonts-noto-cjk yaru-theme-gtk vim-runtime \
     ubuntu-wallpapers-jammy humanity-icon-theme

sudo apt-get autoclean
sudo reboot now
```

0.2.6 Jupyter notebook setup

Here are some instruction on how to install and setup Jupyter on a KRIA board, accessing it remotely and using it for making data analysis.

The following commands will set the required packages and install Jupyter itself[5]:

```
sudo apt-get update && sudo apt-get install python3 python3-pip python3-venv python3-virtualenv

virtualenv myjupyter
source ./myjupyter/bin/activate
python3 -m pip install jupyter pandas numpy matplotlib scipy

sudo reboot now
```

Then in a terminal on your host machine (not on the KRIA board), you can run the following command[6] to bind local ports:

```
ssh -L 8888:localhost:8888 ubuntu@192.168.11.107
```

Then on the opened SSH shell to the KRIA board:

```
source ./myjupyter/bin/activate
jupyter notebook
```

From there, it is possible to use the displayed URL (something that looks like `http://localhost:8888/tree?token`) to access the remote Notebook system from a local web browser. It is possible to do so with `localhost` since we have the `ssh` port map connection going on.

Eventually creating Notebooks and stuff, it is possible to obtain a situation like shown in the figure 1 below.

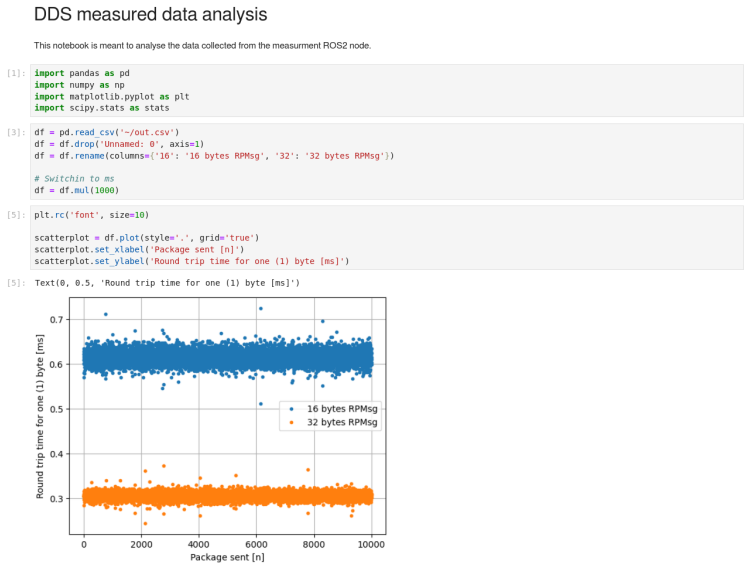

Figure 1: A test Jupyter Notebook for CSV data analysis.

[5] Alongside other packages useful for data analysis, such as `pandas` or `numpy`.

[6] In this example, the full `username@IP` is used, but a `.ssh/config` is aslo usable.

0.2.7 Enabling `remoteproc` with Device-Tree Overlay patching

One of the advantage of this Kria board, as cited previously, is the presence of multiple types of core (APU, MCU, FPGA) on the same chip.

The part in focus in this guide is the usage of both the APU, running a Linux distribution and ROS2; and the MCU, running FreeRTOS and micro-ROS. Online available guides[7] [8] also provide information on how to deploy these types of systems and enabling `remoteproc` for the Kria board, but this guide will show a step-by-step, tried process to have a heterogeneous system up and running.

The communication between both side is meant to be done using shared memory, but some extra setup is required in order to be running the real-time firmware, in particular for deploying micro-ROS on it.

As a first step in that direction, this section of the report will present how to setup and use as an example firmware that utilizes the `remoteproc` device in Linux in order to access shared memory and communicate with the real-time firmware using the RPMsg system.

The communication system and interaction from the Linux side towards the real-time capable core is not enabled by default within the Ubuntu image provided by Xilinx.

In that regard, some modification of the device tree overlay (DTO) is required in order to have the `remoteproc` system starting.

Firstly, we need to get the original firmware device tree, converted into a readable format (DTS):

```
sudo dtc /sys/firmware/fdt 2> /dev/null > system.dts
```

Then, a custom-made patch file can be downloaded and applied. This file is available at the URL visible in the command below but also in this report appendix A.

```
wget
↪ https://gitlab.com/sunoc/xilinx-kria-kv260-documentation/-/raw/b7300116e153f4b5a1542f8804e4646db8030033/src/system.patch

patch system.dts < system.patch
```

As for the board to be able to reserve the correct amount of memory with the new settings, some `cma` kernel configuration is needed[9]:

```
sudo nano /etc/default/flash-kernel

LINUX_KERNEL_CMDLINE="quiet splash cma=512M cpuidle.off=1"
LINUX_KERNEL_CMDLINE_DEFAULTS=""
sudo flash-kernel
```

Now the DTS file has been modified, one can regenerate the binary and place it on the `/boot` partition and reboot the board:

```
dtc -I dts -O dtb system.dts -o user-override.dtb
sudo mv user-override.dtb /boot/firmware/
sudo reboot now
```

After rebooting, you can check the content of the `remoteproc` system directory, and a `remoteproc0` device should be visible, as follow:

[7] A slideshow (JP) from Fixstar employees presents how to use the device tree to enable the communication between the cores.

[8] A blog post (JP) shows all major steps on how to enable the `remoteproc`.

[9] The overlapping memory will not prevent the board to boot, but it disables the PWM for the CPU fan, which will then run at full speed, making noise.

```
ls /sys/class/remoteproc/
#  remoteproc0
```

If it is the case, it means that the patch was successful and that the remote processor is ready to be used!

0.2.8 Installing Docker

It is possible to have a version of Docker installed simply by using the available repository, but since we are on Ubuntu, a PPA is available from Docker in order to have the most up-to-date version.

Following the official documentation, the following steps can be taken to install the latest version of Docker on a Ubuntu system. The last command is meant to test the install. If everything went smoothly, you should see something similar to what is presented in the figure 2 below, after the commands:

```
sudo apt-get update
sudo apt-get install ca-certificates
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
    sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \
    "deb [arch="$(dpkg --print-architecture)" \
  signed-by=/etc/apt/keyrings/docker.gpg] \
  https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && \
        echo "$VERSION_CODENAME")" stable" | \
    sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli \
     containerd.io docker-buildx-plugin docker-compose-plugin
sudo usermod -aG docker $USER
newgrp docker

docker run hello-world
```

```
ubuntu@kria:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
70f5ac315c5a: Pull complete
Digest: sha256:fc6cf906cbfa013e80938cdf0bb199fbdbb86d6e3e013783e5a766f50f5dbce0
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Figure 2: The return of a successful run of the `hello world` test Docker container.

A DTO patch

This file is available in this repository: system.patch

```
diff -u --label /ssh\:kria\:/home/ubuntu/system_original.dts --label /ssh\:kria\:/home/ubuntu/system.dts /tmp/tramp.KJbEbz.dts /tmp/tramp.zHBG7v.dts
--- /ssh:kria:/home/ubuntu/system_original.dts
+++ /ssh:kria:/home/ubuntu/system.dts
@@ -138,7 +138,7 @@

         firmware {

-                zynqmp-firmware {
+                zynqmp_firmware: zynqmp-firmware {
                         compatible = "xlnx,zynqmp-firmware";
                         #power-domain-cells = <0x01>;
                         method = "smc";
@@ -719,7 +719,7 @@
                         phandle = <0x44>;
                 };

-                interrupt-controller@f9010000 {
+                gic: interrupt-controller@f9010000 {
                         compatible = "arm,gic-400";
                         #interrupt-cells = <0x03>;
                         reg = <0x00 0xf9010000 0x00 0x10000 0x00 0xf9020000 0x00 0x20000 0x00 0xf9040000 0x00 0x20000 0x00 0xf9060000 0x00 0x20000>;
@@ -1536,7 +1536,7 @@
                         pinctrl-names = "default";
                         u-boot,dm-pre-reloc;
                         compatible = "xlnx,zynqmp-uart\0cdns,uart-r1p12";
-                        status = "okay";
+                        status = "disabled";
                         interrupt-parent = <0x04>;
                         interrupts = <0x00 0x16 0x04>;
                         reg = <0x00 0xff010000 0x00 0x1000>;
@@ -1909,6 +1909,84 @@
                 pwms = <0x1b 0x02 0x9c40 0x00>;
         };

+       reserved-memory {
+           #address-cells = <2>;
+           #size-cells = <2>;
+           ranges;
+           rpu0vdev0vring0: rpu0vdev0vring0@3ed40000 {
+               no-map;
+               reg = <0x0 0x3ed40000 0x0 0x4000>;
+           };
+           rpu0vdev0vring1: rpu0vdev0vring1@3ed44000 {
+               no-map;
+               reg = <0x0 0x3ed44000 0x0 0x4000>;
+           };
+           rpu0vdev0buffer: rpu0vdev0buffer@3ed48000 {
+               no-map;
+               reg = <0x0 0x3ed48000 0x0 0x100000>;
```

```
+            };
+            rproc_0_reserved: rproc_0_reserved@3ec00000 {
+                no-map;
+                reg = <0x0 0x3ec00000 0x0 0x140000>;
+            };
+        };
+        tcm_0a: tcm_0a@ffe00000 {
+            no-map;
+            reg = <0x0 0xffe00000 0x0 0x15000>;
+            status = "okay";
+            compatible = "mmio-sram";
+            power-domain = <&zynqmp_firmware 15>;
+        };
+        tcm_0b: tcm_0b@ffe20000 {
+            no-map;
+            reg = <0x0 0xffe20000 0x0 0x15000>;
+            status = "okay";
+            compatible = "mmio-sram";
+            power-domain = <&zynqmp_firmware 16>;
+        };
+        rf5ss@ff9a0000 {
+            compatible = "xlnx,zynqmp-r5-remoteproc";
+            xlnx,cluster-mode = <1>;
+            ranges;
+            reg = <0x0 0xFF9A0000 0x0 0x15000>;
+            #address-cells = <0x2>;
+            #size-cells = <0x2>;
+            r5f_0 {
+                compatible = "xilinx,r5f";
+                #address-cells = <2>;
+                #size-cells = <2>;
+                ranges;
+                sram = <&tcm_0a &tcm_0b>;
+                memory-region = <&rproc_0_reserved>, <&rpu0vdev0buffer>,
 ↪ <&rpu0vdev0vring0>, <&rpu0vdev0vring1>;
+                power-domain = <&zynqmp_firmware 7>;
+                mboxes = <&ipi_mailbox_rpu0 0>, <&ipi_mailbox_rpu0 1>;
+                mbox-names = "tx", "rx";
+            };
+        };
+        zynqmp_ipi1 {
+            compatible = "xlnx,zynqmp-ipi-mailbox";
+            interrupt-parent = <&gic>;
+            interrupts = <0 29 4>;
+            xlnx,ipi-id = <7>;
+            #address-cells = <1>;
+            #size-cells = <1>;
+            ranges;
+            /* APU<->RPU0 IPI mailbox controller */
+            ipi_mailbox_rpu0: mailbox@ff990600 {
+                reg = <0xff990600 0x20>,
+                      <0xff990620 0x20>,
+                      <0xff9900c0 0x20>,
+                      <0xff9900e0 0x20>;
```

```
+                reg-names = "local_request_region",
+                        "local_response_region",
+                        "remote_request_region",
+                        "remote_response_region";
+                #mbox-cells = <1>;
+                xlnx,ipi-id = <1>;
+            };
+       };
+
+
          __symbols__ {
                  cpu0 = "/cpus/cpu@0";
                  cpu1 = "/cpus/cpu@1";

Diff finished.  Wed May 24 10:15:49 2023
```