

Advanced Network Security Lab

Session 1: Setup and Network Communication

Dominik Roy George
d.r.george@tue.nl

Savio Sciancalepore
s.sciancalepore@tue.nl

February 21, 2024

1 Introduction

This first lab session provides you with an insight into the reference OS used for lab sessions, namely, *contiki-ng*, as well as how to compile and deploy an application on the *Sensortag CC2650*.

The main objective of this first session is to let you familiarize with the environment, as well as to configure a functional test IoT network where you can ping any device from the device hosting the border router.

For this assignment, we will be working with two example programs of the *contiki-ng*. We need two simple IoT nodes and a border router. The two nodes will be flashed with the *hello-world* program of the *contiki-ng*, while the border router will build on the example program *rpl-border-router*. This document provides the required instructions. We kindly request the students follow the instructions during the lab session.

1.1 Scenario

For the first lab session, you will be working in a **group of three members**. For each group, we will provide three *Sensortags CC2650*. In Fig. 1, we illustrate the scenario you will deploy during the lab session.

Two sensortags will behave as *simple nodes*, and will be communicating with the border-router and with each other. The third sensortag, connected to a Laptop, will behave as the border-router. At the same time, the simple nodes can be connected to the laptops of the other group members.

In Tab. 1.1, we provide *group ids* and associated *PAN IDs* and *Channels*. We provide these ids to prevent mutual interferences during the lab sessions. Therefore, we kindly request to each group not to configure the devices to non-associated ids.

To fully enjoy the first lab session, we provide instructions in four sections: (i) Simple Node Configuration, (ii) Border-Router Configuration, (iii) Communication Setup, and (iv) Wireshark.

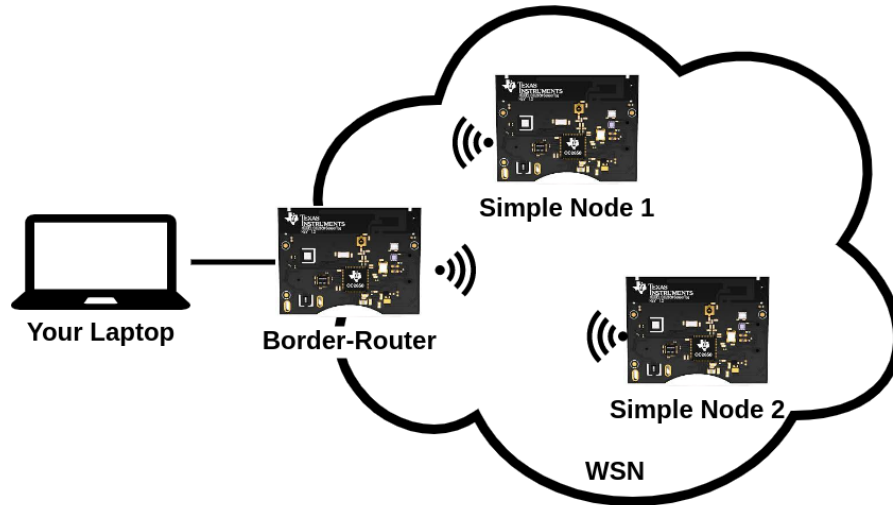


Figure 1: Scenario for Lab Session 1.

2 Simple Node Configuration

You need to apply the following steps for all the *Sensortags CC2650* which act as *simple nodes* in your network.

2.1 Step 1: Configuration

Navigate to the *hello-world* folder and create a *project-conf.h*:

Listing 1: Creation of project-conf.h.

```
$ cd /your-working-folder/contiki-ng/examples/hello-world
#create a file named project-conf.h
$ touch project-conf.h
```

Open the file *project-conf.h* with any editor of your choice. Next, copy the following code inside:


Listing 2: project-conf.h

```
#ifndef PROJECT_CONF_H_
#define PROJECT_CONF_H_

#undef IEEE802154_CONF_DEFAULT_CHANNEL
#define IEEE802154_CONF_DEFAULT_CHANNEL 26
#undef IEEE802154_CONF_PANID
#define IEEE802154_CONF_PANID 0xABCD
```

Group ID	PAN ID	CHANNEL
1	0xABCD	11
2	0xABCE	12
3	0xABCF	13
4	0xABD0	14
5	0xABD1	15
6	0xABD2	16
7	0xABD3	17
8	0xABD4	18
9	0xABD5	19
10	0xABD6	20
11	0xABD7	21
12	0xABD8	22
13	0xABD9	23
14	0xABDA	24
15	0xABDB	25
16	0xABDC	11
17	0xABDD	12
18	0xABDE	13
19	0xABDF	14
20	0xABE0	15
21	0xABE1	16
22	0xABE2	17
23	0xABE3	18

```
#endif
```

 We remind you to change the variable *IEEE802154_CONF_DEFAULT_CHANNEL* and *IEEE802154_CONF_PANID* according to your *group id* in Tab. 1.1.

Afterwards, open the file *Makefile* and add the following line: *MODULES += os/services/shell*. This enables the interactive shell of *contiki-ng*. The *Makefile* should look the following after adding the line:

Listing 3: Makefile.

```
CONTIKI_PROJECT = hello-world
all: $(CONTIKI_PROJECT)

CONTIKI = ../..
## ADD LINE
MODULES += os/services/shell
## END OF LINE
include $(CONTIKI)/Makefile.include
```

2.2 Step 2: Compile

After updating the *project-conf.h*, you need to execute the following command to compile the program, generating an executable file for the *Sensortag CC2650*:

Listing 4: Compile

```
$ sudo make TARGET=cc26x0-cc13x0 BOARD=sensortag/cc2650
```

For verification, if the compilation was successful, you should receive the following output:

Listing 5: Compilation Output

```
.
.
.
CC ../../os/sys/node-id.c
CC ../../os/sys/process.c
CC ../../os/sys/rtimer.c
CC ../../os/sys/stack-check.c
CC ../../os/sys/stimer.c
MKDIR build/cc26x0-cc13x0/sensortag/cc2650/gen
CC ../../arch/cpu/cc26x0-cc13x0/./ieee-addr.c
CC ../../arch/cpu/cc26x0-cc13x0/lib/cc26xxware/startup_files/
    ccfg.c
CC ../../os/contiki-main.c
CC hello-world.c
LD build/cc26x0-cc13x0/sensortag/cc2650/hello-world.elf
CP build/cc26x0-cc13x0/sensortag/cc2650/hello-world.elf -->
    build/cc26x0-cc13x0/sensortag/cc2650/hello-world.cc26x0-
    cc13x0
CP build/cc26x0-cc13x0/sensortag/cc2650/hello-world.cc26x0-
    cc13x0 --> hello-world.cc26x0-cc13x0
```

After compiling copy the executable into your shared folder of the Virtual Machine (VM) (e.g.: ans/sf_assignments).

2.3 Step 3: Flash

Initiate the UNIFLASH program through the following:

Listing 6: Starting UNIFLASH

```
#Navigate to your UNIFLASH folder:
$ cd ~/ti/uniflash_8.0.0/
# Start UNIFLASH:
$ ./uniFlashGUI.sh
```

After starting the UNIFLASH, you have to select the BOARD: *CC2650F128 On-Chip* in the tab *New Configurations*, shown in Fig. 2. After selecting the

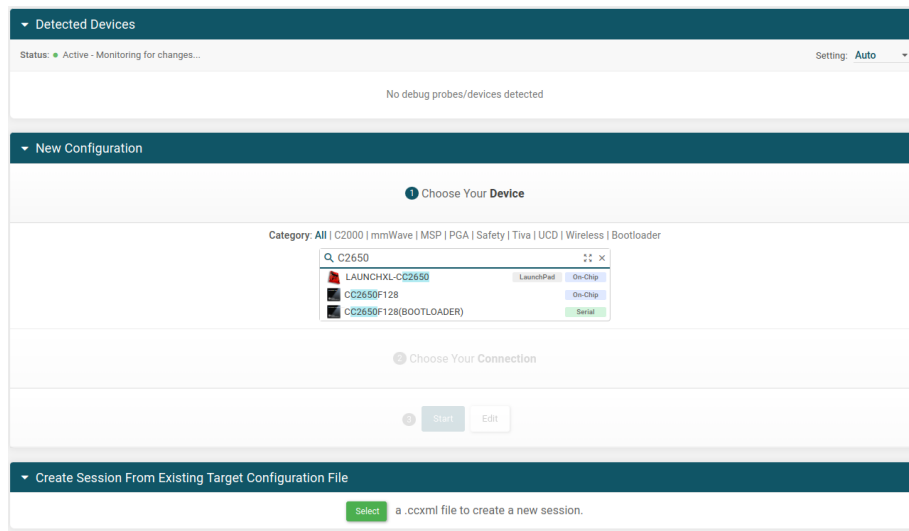


Figure 2: Board Selection.

board, you have to select the debugger *Texas Instruments XDS110 USB Debug Probe*, shown in Fig. 3.

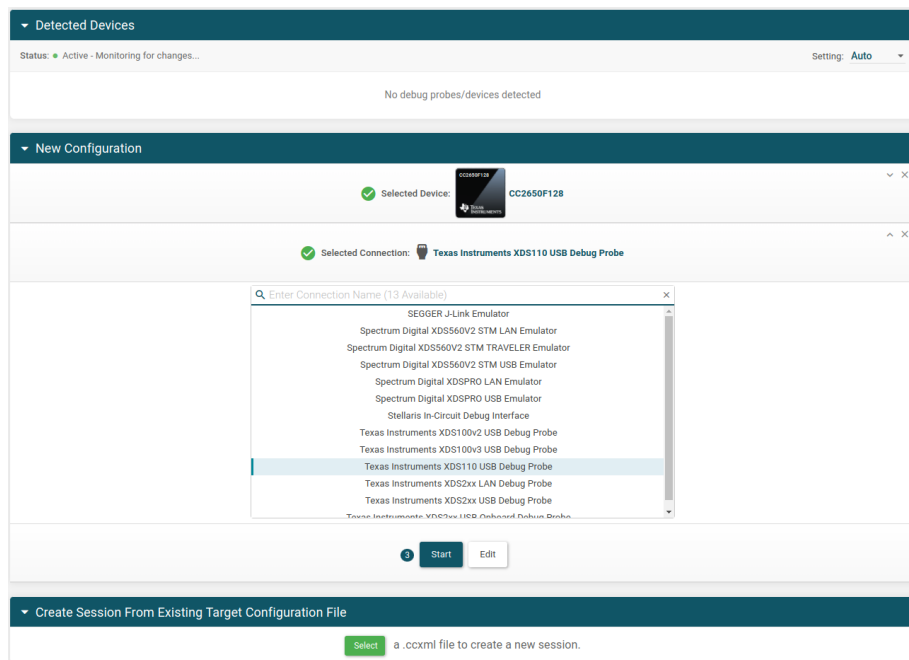


Figure 3: Debugger Selection.

You have to select the executable by clicking the *Browse* button. At the

same time, to flash the executable on the device (the executable is located in the base folder of the project, *hello-world.cc26x0-cc13x0*), click on the *Load Image* button, shown in Fig. 4.

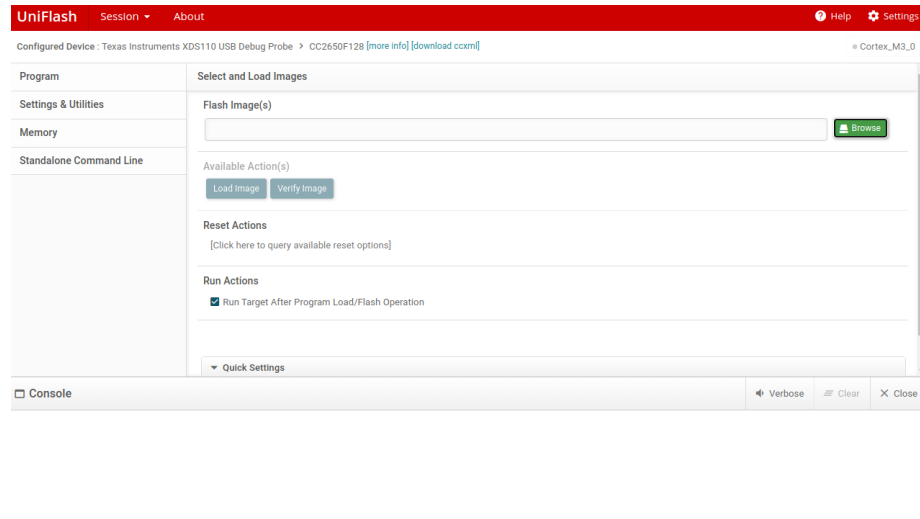


Figure 4: Load Executable

After flashing the executable on the device, you have to reset the device by choosing *Board Reset* in the *Reset Actions* Section and then you have to click on the *Reset Now* Button, shown in Fig. 5.

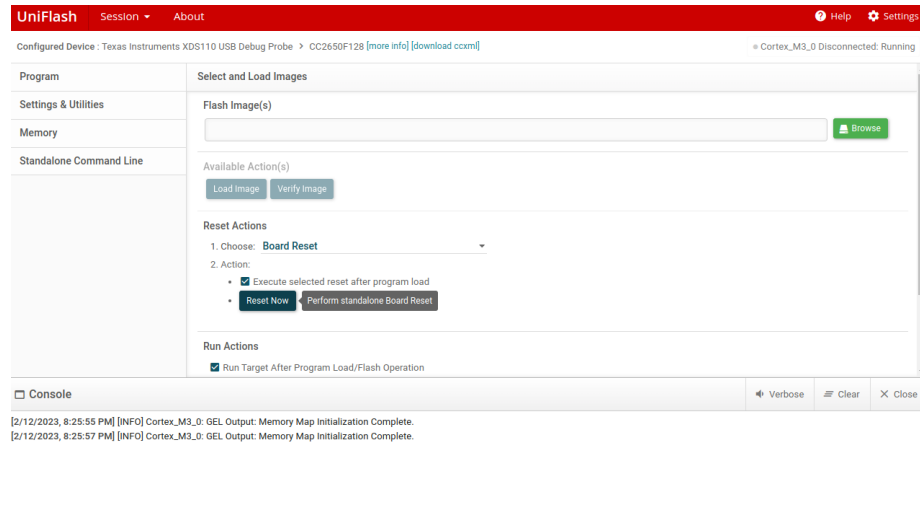


Figure 5: Reset Device.

2.4 Step 4: Execute

First connect the sensortag to your host machine. Next, pipe the USB connection to your VM as shown in Fig. 6 . You should be able to see the Sensortag

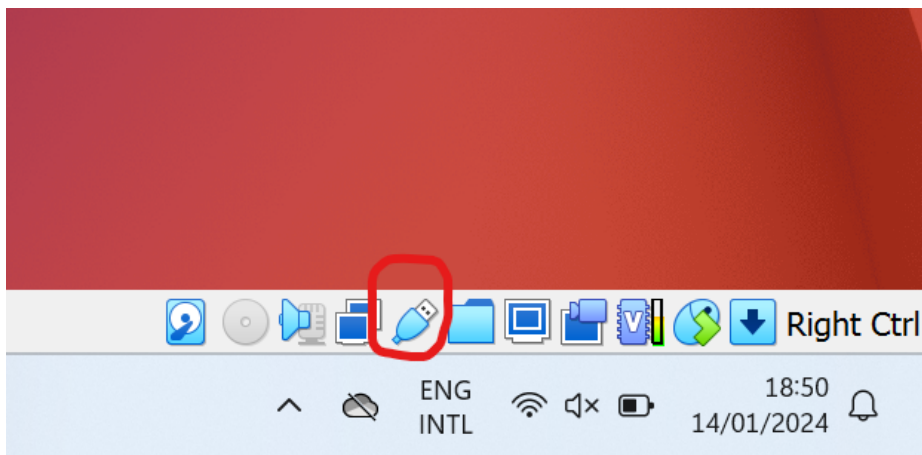


Figure 6: VM USB ICON.

listed in the available USB connections as shown in Fig. 7.

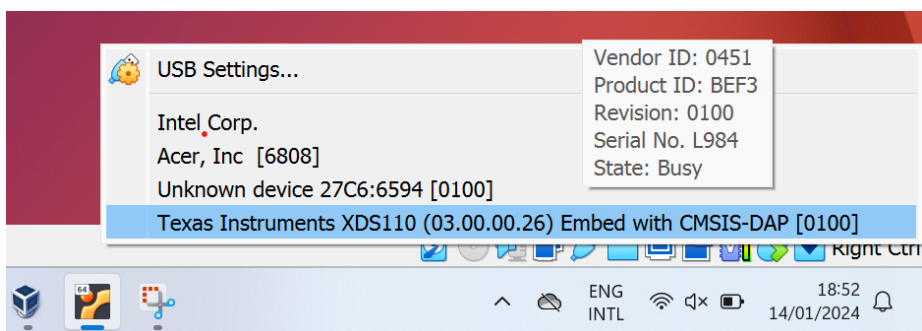


Figure 7: VM USB Selection.

Click on the Sensortag and if it was successfully connect, then you should be able to see a check next to as shown in Fig. 8.

To confirm that the USB serial exists you can type the command as shown in Fig. 9.

To verify that the previous steps were executed and completed successfully, you have to execute the following command on the terminal, which logs into the device:

Listing 7: Command for Verification.

```
$sudo rlwrap ../../tools/serial-io/serialdump -b115200 /dev/  
ttyACM0
```

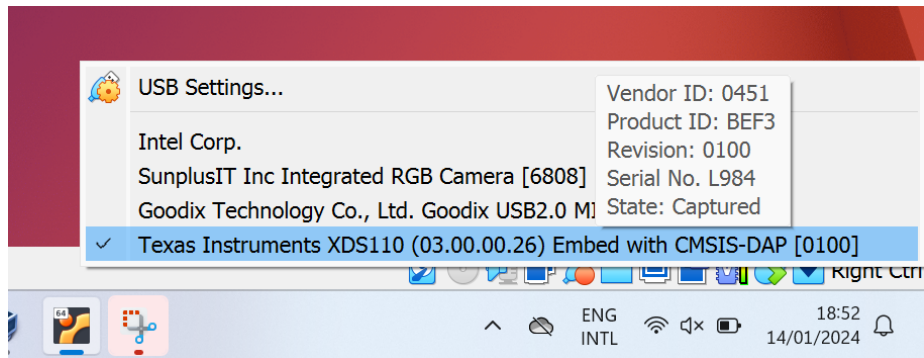


Figure 8: VM USB Selected.

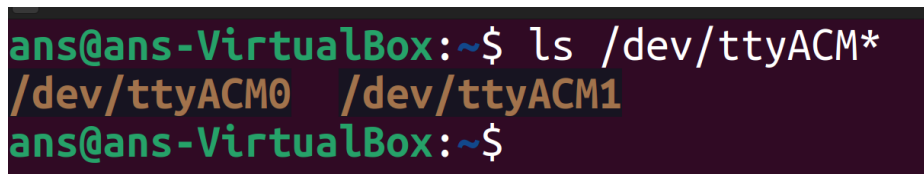


Figure 9: VM USB Connected.

To verify the output, we provide an example excerpt in Fig. 10.

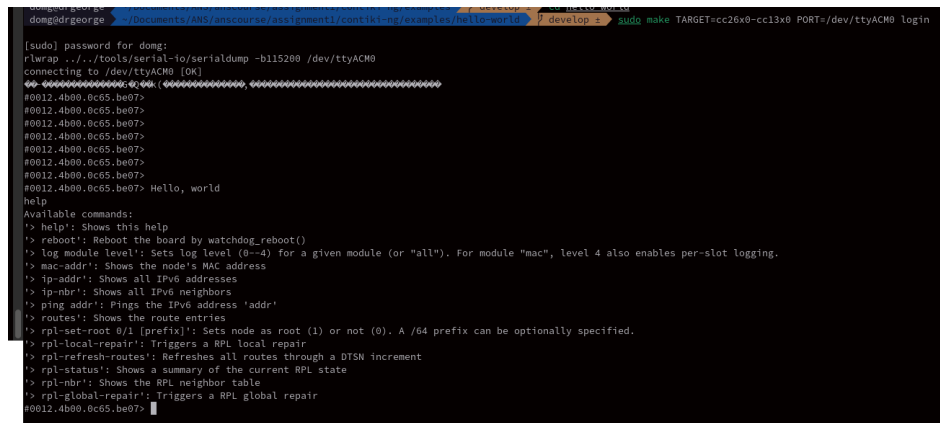



Figure 10: Verification Output.

 We remark that, if you cannot login or you do not see the output, unplug the device and plug the device into USB again!

3 Border-Router Configuration

In this lab session, we use a border-router to route data between an IoT network (RPL network) and an external IPv4 network. For more details, we refer to [1, 2].

3.1 Step 1: Configuration

Navigate to the *rpl-border-router* folder and edit the *project-conf.h*:


Listing 8: RPL-Border-Router.

```
$ cd /your-working-folder/contiki-ng/examples/rpl-border-router
```

Open the file *project-conf.h* with any editor of your choice and copy the following code inside:

Listing 9: project-conf.h

```
#undef IEEE802154_CONF_DEFAULT_CHANNEL
#define IEEE802154_CONF_DEFAULT_CHANNEL YOUR_CHANNEL
#undef IEEE802154_CONF_PANID
#define IEEE802154_CONF_PANID YOUR_PAN_ID
```

 We remind you to change the variable *IEEE802154_CONF_DEFAULT_CHANNEL* and *IEEE802154_CONF_PANID* according to your *group id* in Tab. 1.1.

3.2 Steps 2 and 3

Steps 2 and 3 are equivalent as presented in Sec. 2.

3.3 Step 4: Execute

First connect the sensortag to your host machine. Next, pipe the USB connection to your VM as shown in Fig. 11. You should be able to see the Sensortag listed in the available USB connections as shown in Fig. 12.

Click on the Sensortag and if it was successfully connect, then you should be able to see a check next to as shown in Fig. 13.

To confirm that the USB serial exists you can type the command as shown in Fig. 14.

To verify that the previous steps were executed successfully, execute the following command on the terminal, which logs into the device:

Listing 10: Command for Verification.

```
$ sudo rlwrap ../../tools/serial-io/serialdump -b115200 /dev/
ttyACM0
```

To verify the output we provide an example excerpt, shown in Fig. 15.

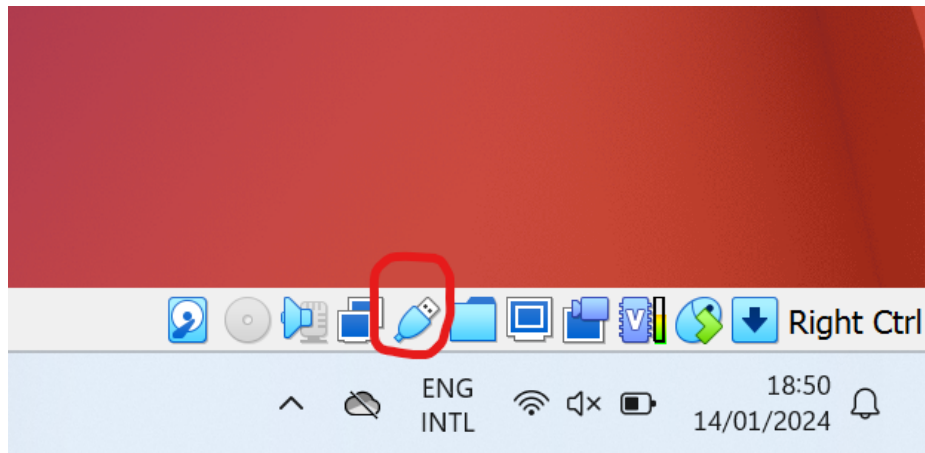


Figure 11: VM USB ICON.

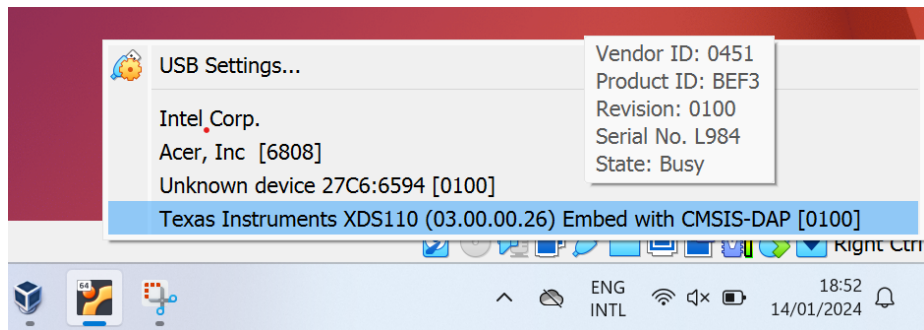


Figure 12: VM USB Selection.

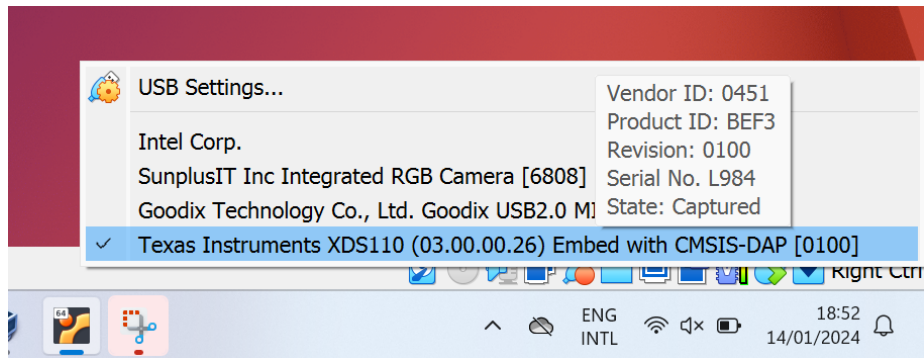


Figure 13: VM USB Selected.

👉 We remark that, if you cannot login or you do not see the output, unplug the device and plug the device into USB again!

Next, you have to open a browser of your choice and write **the IPv6 address of your border-router**, such as `http://[fd00::212:4b00:c4a:6c87]/`. If your operation is successful, you should have an output similar to the one in Fig. 17.

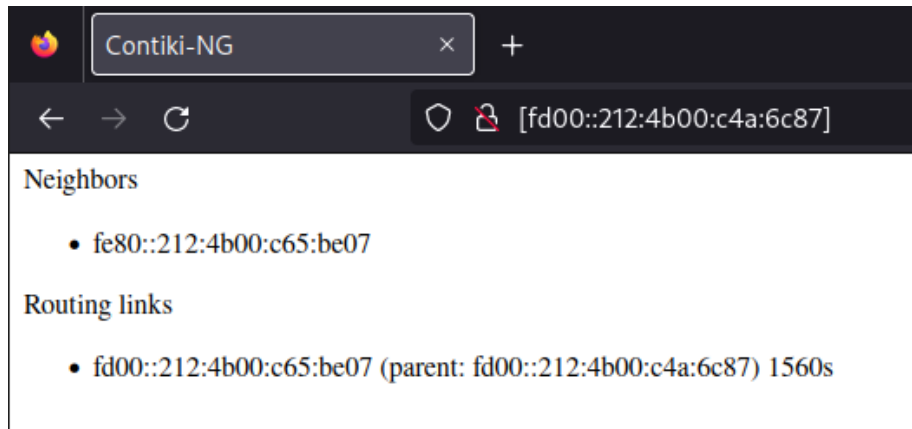


Figure 17: Browser Output.

Following, you should be able to ping from your host system all the devices in your local network, by using their IPv6 address, shown in Fig. 18.

```
domg@drgeorge ~$ ping -6 fd00::212:4b00:c4a:6c87
PING fd00::212:4b00:c4a:6c87(fd00::212:4b00:c4a:6c87) 56 data bytes
64 bytes from fd00::212:4b00:c4a:6c87: icmp_seq=1 ttl=64 time=19.4 ms
64 bytes from fd00::212:4b00:c4a:6c87: icmp_seq=2 ttl=64 time=19.6 ms
64 bytes from fd00::212:4b00:c4a:6c87: icmp_seq=3 ttl=64 time=19.6 ms
^C
--- fd00::212:4b00:c4a:6c87 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 19.416/19.534/19.635/0.090 ms
domg@drgeorge ~$ ping -6 fd00::212:4b00:c65:be07
PING fd00::212:4b00:c65:be07(fd00::212:4b00:c65:be07) 56 data bytes
64 bytes from fd00::212:4b00:c65:be07: icmp_seq=1 ttl=63 time=68.0 ms
64 bytes from fd00::212:4b00:c65:be07: icmp_seq=2 ttl=63 time=83.6 ms
64 bytes from fd00::212:4b00:c65:be07: icmp_seq=3 ttl=63 time=99.2 ms
^C
--- fd00::212:4b00:c65:be07 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 68.042/83.606/99.183/12.713 ms
```

Figure 18: Ping Output.

5 Wireshark Test

To capture the traffic from the border-router, you have to open wireshark and capture all the traffic of the interface *tun0*, as shown in Fig.19.

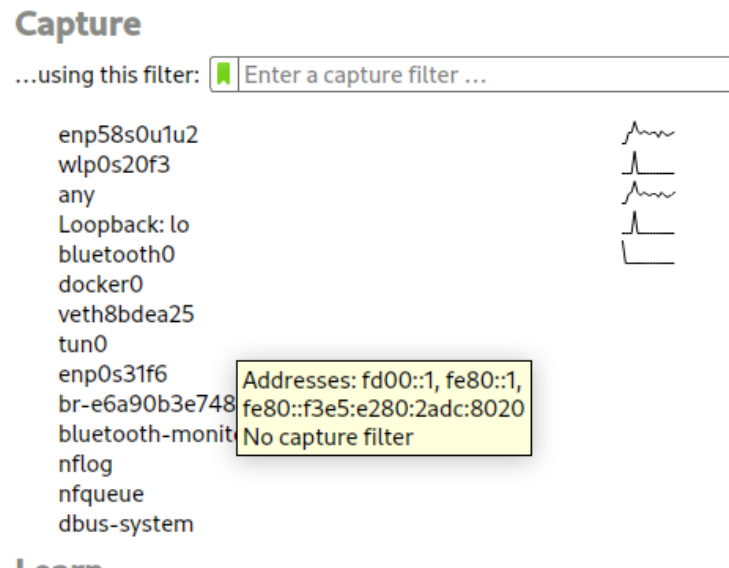


Figure 19: Wireshark Interface Selection.

Afterwards, you should visualize the traffic of the border-router, shown in Fig. 20.

Take some time to familiarize with the content of the logged packets, trying to recognize the protocols and headers discussed in the frontal lectures.

If all the previous steps completed successfully, then you finished successfully the first lab session.

You can also have a look around in the code of *contiki-ng* to familiarize with the several components of the protocol stack and the available examples.

References

- [1] Contiki-NG, “RPL Border Router.” [Online]. Available: <https://docs.contiki-ng.org/en/develop/doc/tutorials/RPL-border-router.html>
- [2] ANRG, “Contiki Tutorials.” [Online]. Available: https://anrg.usc.edu/contiki/index.php/Contiki_tutorials

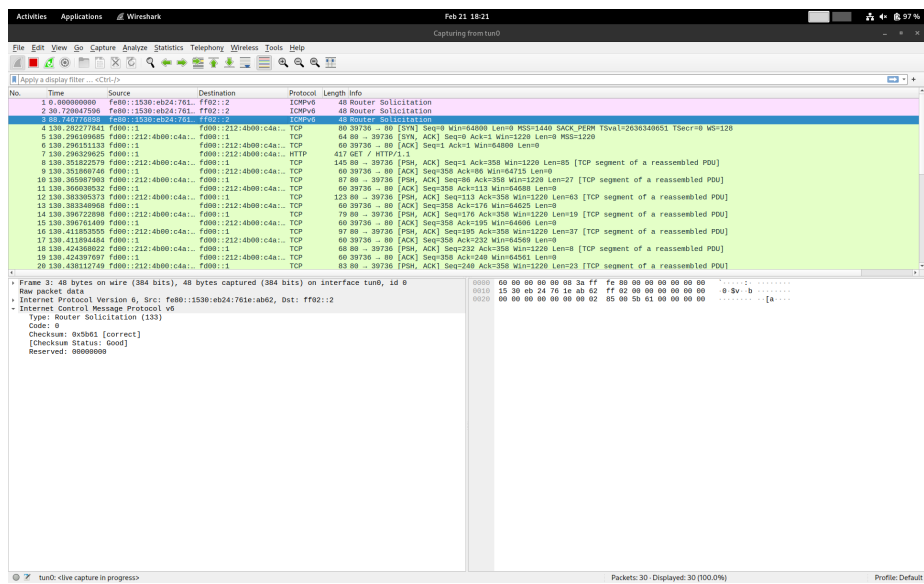


Figure 20: Wireshark Output.